Universitat Politècnica de Catalunya

Universitat de Barcelona

Grau en Estadística

Treball de Final de Grau

Models generatius d'aprenentatge profund: Estudi del Autoencoder Variacional

Adrián Vizoso Expósito

Setembre 2020

Tutor: Esteban Vegas Lozano

Facultat de Matemàtiques i Estadística

Facultat d'Economia i Empresa

Resum

L'**autoencoder variacional** es tracta d'un model d'aprenentatge profund que parteix de la clàssica estructura dels autoencoders per tal de generar noves dades. Aquest procés és possible gràcies a emular el comportament del cervell humà, projectant les dades en un espai latent on es recullen les característiques principals de les dades introduïdes de forma similar com ho fem nosaltres al utilitzar la nostra imaginació.

S'introdueixen en primer lloc tots aquells conceptes necessaris per entendre el funcionament d'aquests models. Es parla sobre xarxes neuronals i com podem pensar en aquestes com models probabilístics, es detallen les mesures més importants emprades en el treball des de l'òptica de la teoria de la informació per tal de quantificar i comparar distribucions de probabilitat i per últim es presenta una taxonomia general dels models generatius.

A continuació s'explica amb tot detall com funcionen els autoencoders variacionals. S'explica en què consisteixen exactament, quin paper té la funció objectiu alhora d'entrenar aquests models i quina és la seva interpretació.

Durant el projecte, s'implementen dos autoencoders variacionals amb dos estructures de capes diferents. Aquets models s'implementen utilitzant els softwares de programació *Python* i R on es fa l'ús de les llibreries *Tensorflow* i *Keras*.

Seguidament, a través de la hiperparametrització, es troba el conjunt de paràmetres que permeten generar imatges amb la millor qualitat possible per aquests models i es generen algunes a mode d'exemple.

Després de reflexionar sobre el comportament d'aquets models tan singulars, es modifica la funció objectiu per tal de dur a terme la transició cap al *MMD-VAE*. Un enfoc alternatiu que també permet la generació de noves imatges.

El treball culmina amb un apartat de conclusions on es recullen les idees més importants del treball i s'exposen els resultats obtinguts durant la recerca.

Paraules clau: *Autoencoder Variacional*, *Models Generatius*, *Inferència Variacional*, *Conv-VAE*, *Tensorflow*, *Keras*, *Models de variable latent*, *MMD-VAE*

Abstract

The **variational autoencoder** is a deep learning model that starts from the classic structure of autoencoders in order to generate new data. This process is possible by emulating the behavior of the human brain, projecting the data into a latent space where the main characteristics of the input data are collected in a similar way as we do when using our imagination.

During the project two variational autoencoders are implemented by changing the internal layer structure. These autoencoders are implemented in Python and R using together Tensorflow and Keras libraries.

In this project, besides this, a new approach of this model is introduced: the MMD-VAE. Modifying the loss function, we can obtain a model that is able to achieve good results that are worth to look at.

To conclude the thesis, we summarize all the results and insights obtained from the research.

Keywords: *Variational Autoencoder*, *Generative models*, *Conv-VAE*, *Variational Inference*, *Tensorflow*, *Keras*, *Latent variable models*, *MMD-VAE*

Agraïments

Al meu tutor Esteban, per tot el suport i ajuda durant tot aquest temps.
A la meva família i amics, per estar sempre al meu costat.

Índex

Llistat de Figures

Llistat de Taules

Llista d'abreviacions

AI	Inteligència Artificial
ML	Machine Learning
DL	Deep Learning
MLP	Perceptró Multicapa
CNN	Xarxa Neuronal Convulucional
DNN	Xarxa Neuronal Profunda
AE	Autoencoder
VAE	Autoencoder Variacional
IV	Inferència Variacional
KL	Divergència de Kullback-Leibler
ReLU	Rectified Linear Unit

Conceptes previs

Capítol I

Les xarxes neuronals

1.1 Conceptes claus

Amb l'expansió durant les últimes dècades del món de la intel · ligència artificial on quasi cada dia escoltem aquesta paraula al nostre entorn. Una gran confusió s'ha introduït en aquests conceptes. És per això que abans de començar a aprofundir en el nostre tema principal s'hi intenten donar unes definicions acotades dels conceptes que engloben l'àrea principal del treball consolidant i conciliant les idees bàsiques.

1.2 Intel · ligència artificial

El concepte d'intel · ligència artificial sembla un concepte molt ample i difícil d'acotar. En aquest treball es proposa una definició que busca entendre aquest terme des d'un punt de vista simple sense entrar en ambigüitats. *Definim per tant d'intel · ligència artificial com qualsevol procés o tècnica que busqui la replicació del pensament humà.* El pensament humà al qual es fa referència en la definició s'entén com la capacitat d'anàlisis i presa de decisions pròpies dels éssers humans.

Quan es parla d'aquest concepte des d'un punt de vista teòric és fàcil pensar que s'està parlant d'algun tipus de relat de ciència-ficció. La realitat és que la majoria d'aplicacions de la intel · ligència artificial com la detecció d'objectes, la conducció automàtica de vehicles, reconeixement de veu i altres sorprenents aplicacions es basen en complexes tècniques d'algoritmes de *Machine Learning*. Aquest tipus d'algoritmes es detallen a continuació.

1.3 *Machine Learning*

L'aprenentatge automàtic és l'ús de procediments matemàtics (algoritmes) per analitzar dades. L'objectiu és descobrir patrons útils (relacions o correlacions) entre diferents ítems de dades.
Un cop identificades les relacions, aquestes es poden utilitzar per fer inferències sobre el comportament de nous casos quan es presenten. En essència, això és anàleg al que aprenen les persones. Observem què passa al nostre voltant i traiem conclusions de les nostres experiències sobre el funcionament del món. A continuació, apliquem el que hem après per ajudar-nos a afrontar noves situacions en què ens trobem. Com més experimentem i aprenguem, millor serà la nostra capacitat de prendre decisions.

1.4 Deep learning

Definim el *deep learning* com el conjunt de tècniques i algoritmes basats en el *Machine Learning* que tenen com a objectiu extreure aprendre o aproximar la relació o patrons ocults en les dades tant de forma supervisada com no supervisada.

En un principi podria semblar complicat discernir entre aquesta i la definició anterior, però el punt clau entre aquestes dues definicions és comprendre que els mètodes utilitzats en el *deep learning* es basen en una estructura molt concentra com és la xarxa neuronal a diferència del *Machine Learning* que empra mètodes més diversos com són els *SVM* o la regressió logística.
És interessat però remarcar que es tracten de conceptes niats tal com s'observa a la següent figura:

Els algoritmes de *deep learning* funcionen per capes neuronals que simulen de forma simplificada les capes neuronals que formen el cervell humà. Encara que tots els algoritmes de *deep learning* comparteixen aquest tret, l'arquitectura de cadascun d'ells i el seu funcionament és significament diferent. Quan es demostra que una estructura és especialment útil, és freqüent assignar-li un nom per tal d'identificar-la fàcilment.

Durant aquest treball ens centrarem en un tipus d'estructures molt concretes: els models generatius. Aquests models són una aplicació del *deep learning* els quals tracten de generar noves dades extreien abans la distribució de les nostres dades per tal d'introduir petites variacions en aquestes. Encara així, les aplicacions del *deep learning* són molt més extenses i engloben una gran varietat d'àrees.

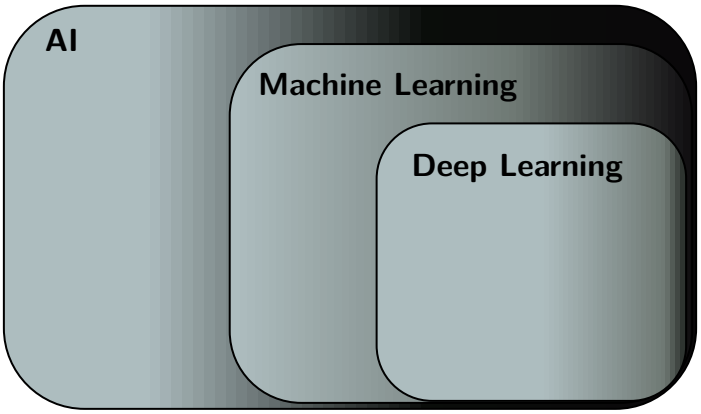


Figura 1.1: Subdivisions de la inteligència artificial
[Font: Elaboració propia]

1.5 Introducció a les xarxes neuronals profundes

1.5.1 Inicis de les xarxes neuronals: el perceptró

Les xarxes neuronals són un conjunt d'algoritmes que tracten d'emular el funcionament del cervell humà a través d'estructures similars. Les unitats bàsiques d'aquests sistemes són les neurones. Aquestes estan interconnectades i organitzades en diferents capes. L'objectiu de simular aquesta arquitectura neuronal, és assolir tasques tan complicades com poden ser la predicció o la classificació d'un conjunt de valors.

La primera xarxa neuronal va ser desenvolupada el 1943 per Warren McCulloch, un neuròleg de la Universitat d'Illinois i Walter Pitts, matemàtic de la Universitat de Chicago. Encara així, donada la falta de recursos tecnològics, no va ser fins a 1954 quan es va aconseguir provar i observar la capacitat d'aquesta xarxa.

Inspirat en aquest model, Frank Rosenblatt va desenvolupar un tipus de neurones artificials anomenades perceptró. Aquest seria un gran avanç que determinaria transcurs d'aquest tipus d'estructures neuronals i és un punt del qual és interessant partir per comprendre l'arquitectura d'una xarxa.

El perceptró de Rosenblatt rebia una sèrie de variables d'entrada dicotòmiques i retornava una sortida també binària. Cada una d'aquestes variables d'entrada tenien associat un pes wi que representaven la importància de cadascuna de les variables. La sortida de la neurona seria 0 o 1 si la suma ponderada $\sum_j wjxj$ superava un determinat llindar. Igual que els pesos, aquest llindar era un paràmetre de la neurona que calia fixar

abans.
:

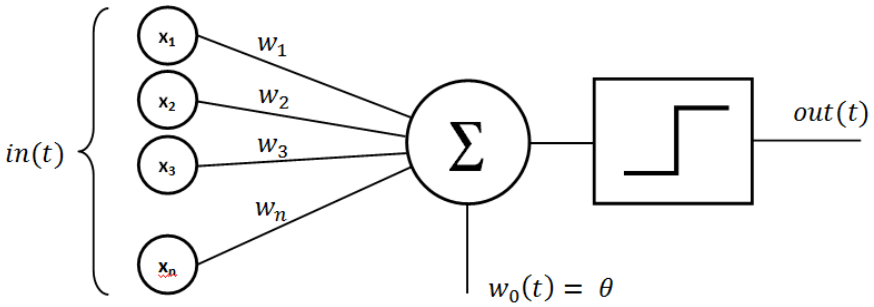


Figura 1.2: Esquema d'un perceptró de sortida binària.
(Font: Mayranna 2013)

Rosenblatt va demostrar que si interconnectàvem aquestes neurones podíem ser capaços de resoldre un problema de classificació binària (7.6) on les dues classes són lineament separables a partir d'un algoritme relativament senzill que es basa a corregir els pesos i el llindar quan cometem un error de classificació fent que l'algoritme convergís i entrenant així el nostre model.

Encara així, condicionades a la potència dels ordinadors de l'època i la dificultat de reproduir aquestes en un entorn diferent del d'un laboratori, les xarxes neuronals van tenir un impacte reduït i molta gent en aquell moment no va confiar en els resultats mostrats.

A partir d'aquest punt, la història del *deep learning* no ha deixat d'avançar contínuament fins a l'actualitat, on es publiquen una gran quantitat d'articles científics i els nous models s'apliquen en una gran quantitat d'àrees.

1.5.2 El ressorgiment de les xarxes neuronals

Gràcies a aquests avenços tecnològics, les xarxes neuronals han pres un paper molt rellevant en l'actualitat. És fàcil associar el ressorgiment actual d'aquestes tècniques a tres grans pilars que han definit aquest nou període.

- *Big Data.*
  Els conjunts de dades massius així com la capacitat actual per emmagatzemar i gestionar aquests conjunts de dades a través de plataformes i softwares dedicats hi ha permet l'avanç i millora de les tècniques utilitzades al món del Deep Learning.

- *Hardware.*
  Millores en les GPU i introducció de la paral · lelització massiva.

- *Software.*
  Tècniques millorades, creació nous models, combinació de les llibreries Keras i TensorFlow de les que es parlarà en detall més endavant.

1.5.3 Conceptes bàsics de les xarxes neuronals: estructures i composició

Tal com s'ha introduït anteriorment, aquestes xarxes es configuren a partir de neurones, la unitat bàsica dins d'una xarxa i l'element que fa possible l'aprenentatge.

Les neurones tenen associat un paràmetre que actua com regulador de la influència que tenen els valors d'entrada de les altres neurones que formen la xarxa: els pesos. Aquests paràmetres determinen si la neurona estarà activada o no.

Quan apilem un conjunt de neurones parlem d'una capa neuronal. Totes les xarxes es contenen una capa neuronal d'entrada, una capa neuronal de sortida i almenys una capa central. Les xarxes que contenen més d'una capa neuronal central, s'anomenen xarxes neuronals profundes: són aquelles xarxes que tenen una capacitat deductiva més alta i poden arribar a modelar relacions més complexes entre variables.

1.5.4 Connexions entre neurones

Per tal de poder connectar cada una d'aquestes neurones necessitem realitzar dues tasques senzilles que modificaran el comportament de la nostra xarxa.

D'una forma similar al perceptró de Rosenblatt, la primera fase es basa en la propagació a través l'activació neuronal: aquest procés es caracteritza per processar una sèrie d'inputs (que poden provenir d'una variable d'entrada o una altra neurona de la xarxa) i retornar un valor numèric que serà propagat a la resta de neurones.

La segona fase s'anomena retropropagació: és el procés on ajustem els pesos w_j associats a cada una d'aquestes neurones. Aquest paràmetre s'ajusta en funció l'error que comès quan comparem la sortida de la xarxa amb el valor que volem predir i és l'element

central que ens permet entrenar la nostra xarxa.

1.5.5 Activació neuronal

Necessitem definir una expressió que ens permeti quantificar l'efecte de les variables d'entrada en aquesta fase d'activació. Utilitzarem per tant una funció $\mathcal{F}_k$ que rep el conjunt de senyals d'entrada total $s_k(t)$, el valor d'activació actual i retorna el valor d'activació per la següent neurona $y_k(t+1)$.

Aquesta expressió serà de la forma:

$$y_k(t+1) = \mathcal{F}_k\left(y_k(t), s_k(t)\right) \tag{1.5.1}$$

És freqüent reescriure aquesta expressió únicament en funció del valor d'entrada de la següent manera:

$$y_k(t+1) = \mathcal{F}_k\left(s_k(t)\right) = \mathcal{F}_k\left(\sum_j w_{jk}(t) y_j(t) + \theta_k(t)\right) \tag{1.5.2}$$

On $thetha_j$ i w_j són les unitats de biax i els pesos per la neurona j, respectivament.

A diferència del perceptró, en les xarxes neuronals busquem una funció $\mathcal{F}_k$ no decreixent per tal d'aportar més flexibilitat en el valor de sortida. Aquestes funcions acostumen a retornar un valor dins d'un interval $[a, b]$. L'exemple més conegut de funció d'activació és la funció sigmode, que es defineix formalment així:

$$y_k = \mathcal{F}\left(s_k\right) = \frac{1}{1 + e^{-s_k}} \tag{1.5.3}$$

Tot i que existeixen altres funcions d'activacions que poden ser utilitzades en funció de les nostres dades a modelar.

1.5.6 Correcció dels paràmetres w_j

Aquesta és la fase que coneixem com propagació cap enrere: un cop hem definit l'estructura de la nostra xarxa (quantes capes tindrà, quantes neurones inclourem en cada capa) serem capaços de generar un output. Aquest primer *output*, serà definit pels valors inicials

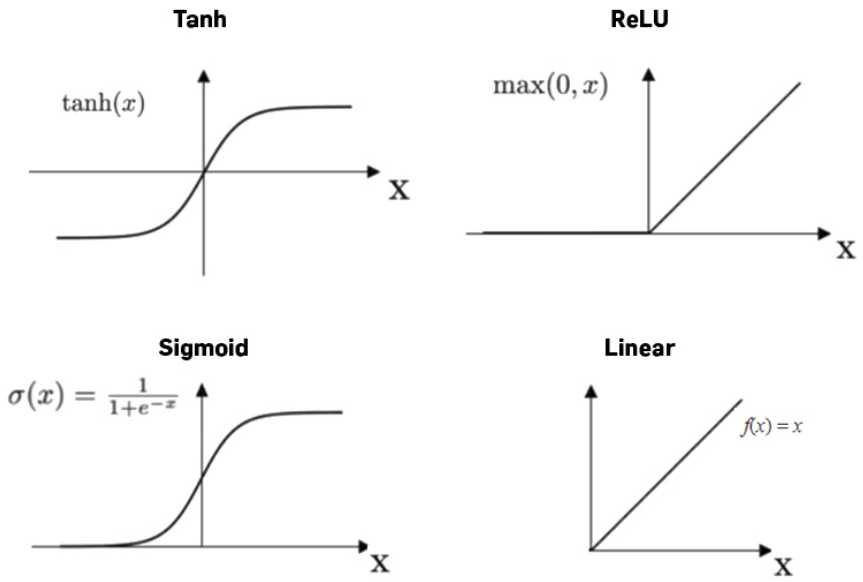


Figura 1.3: Diferents funcions d'activació.
(Font: 2020)

donats als paràmetres de la nostra xarxa. És en aquest moment quan compararem la sortida de la nostra xarxa $\hat{y}$ amb el valor desitjat y. La diferència entre aquests dos valors serà l'error comès en aquesta iteració utilitzant el paràmetre $\vec{w} = \{w_1, \ldots, w_j, \ldots, w_N\}$ i $\vec{\theta} = \{\theta_1, \ldots, \theta_j, \ldots, \theta_N\}$ per un determinat registre x del nostre conjunt de dades l'anomenarem error.

$$e(x, \vec{w}, \vec{\theta}) = y(x) - \hat{y}(x, \vec{\theta}, \vec{w}) \tag{1.5.4}$$

Per tal d'optimitzar els paràmetres per tots els registres que conformen el nostre *dataset* utilitzarem la següent expressió:

$$C(\vec{w}, \vec{\theta}) \equiv \frac{1}{2n} \sum_x \|e(x, \vec{w}, \vec{\theta})\|^2 \tag{1.5.5}$$

A partir d'ara abandonem la notació vectorial pels paràmetres θ i w i passarem a parlar del conjunt de paràmetres $\Theta = \{\vec{w}, \vec{\theta}\}$ de la nostra xarxa per tal de simplificar les equacions següents.

Observem ara que la nostra funció $C(\Theta)$ és una funció no negativa que prendrà valors propers a zero sempre que $y(x) \approx \hat{y}(x, \Theta)$ i valors grans quan $y(x)$ sigui molt diferent de $\hat{y}(x, \Theta)$, buscarem llavors un conjunt de paràmetres Θ que permeti que les nostres prediccions $\hat{y}(x, \Theta)$ siguin tan similars a $y(x)$ com sigui possible.

1.5.7 Optimitzadors

Els optimitzadors són les eines que ens permeten ajustar els paràmetres de la nostra xarxa neuronal a través de la minimització de la funció de costos d'aquesta i que ens permetran resoldre el problema d'optimització plantejat anteriorment. Els diferents optimitzadors són els encarregats de minimitzar la pèrdua. Realitzar aquesta operació, per tant, implicarà que el nostre model sigui capaç d'extreure relacions complexes entre variables.

L'optimitzador més habitual es tracta del descens del gradient. És l'algoritme més bàsic però també un dels més utilitzat donat que amb una parametrització correcta podem obtenir uns resultats més que satisfactoris. Descens del gradient (DG)

El descens del gradient parteix de la derivada de la funció de cost que indica la variació que s'ha d'aplicar en els pesos actuals per tal d'obtenir uns nous actualitzats que minimitzen la funció de cost. L'expressió formal és:

$$\Theta = \Theta - \eta \cdot \nabla_{\Theta} C(\Theta) \tag{1.5.6}$$

On η representa la tassa d'aprenentatge, una hiperparàmetre del model que necessita ser ajustat correctament per tal d'evitar caure en un mínim local. Gràficament, el procés es pot representar de la següent forma si ho pensem en $\mathbb{R}^{\not\vDash}$

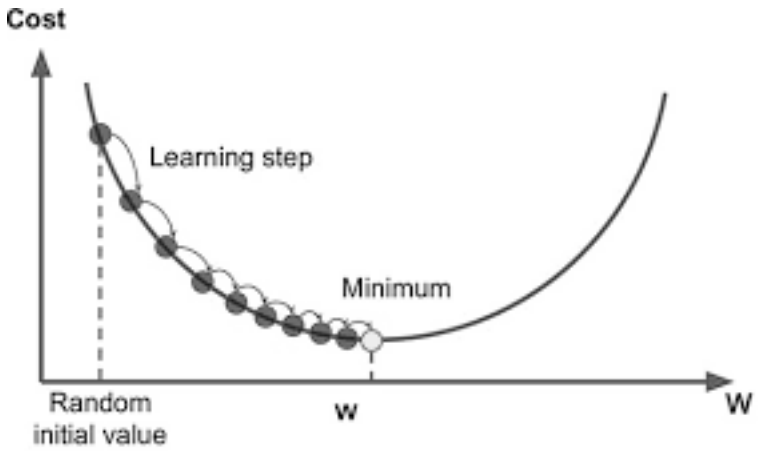


Figura 1.4: Algoritme del descens del gradient representat en $\mathbb{R}^{\not\vDash}$.
(Font: Bhattarai 2018)

El principal inconvenient d'aquest mètode és que els pesos són actualitzats un cop

s'ha iterat sobre tot el conjunt de dades, això fa que aquest mètode sigui especialment costós computacionalment i en la pràctica sovint presenta problemes d'espai de memòria quan la quantitat de registres és gran.

1.5.7.1 Descens del gradient estocàstic (SDG)

Per tal d'evitar aquests potencials problemes es va desenvolupar l'algoritme del descens del gradient estocàstic. Aquest mètode replica la idea anterior amb una diferència significativa: els pesos són actualitzats cada interacció en comptes de després d'iterar sobre el *dataframe*.

Si suposem que estem davant d'un problema d'aprenentatge supervisat on $x^{(i)}$ és un registre del nostre conjunt de dades i $y^{(i)}$ representa l'etiqueta del registre, definim el descens del gradient estocàstic de la següent manera:

$$\Theta = \Theta - \eta \cdot \nabla_{\Theta} C\left(\Theta; x^{(i)}; y^{(i)}\right) \tag{1.5.7}$$

Malauradament, la convergència de l'SDG no sempre està garantida, ja que aquest algorisme és propens al *overshooting* fent difícil assolir el màxim global de la funció de cost.

1.5.7.2 Descens del gradient del mini-batch

La solució definitiva proposada va ser la següent: en comptes d'actualitzar els pesos en cada iteració o després d'iterar sobre totes les dades, actualitzar-los un cop s'hagin assolit n iteracions.

$$\Theta = \Theta - \eta \cdot \nabla_{\Theta} C\left(\Theta; x^{(i:i+n)}; y^{(i:i+n)}\right) \tag{1.5.8}$$

D'aquesta forma aconseguim un algoritme que a través de reduir la variància a l'hora d'actualitzar els pesos, és capaç de convergir d'una forma més consistent que els anteriors. Els valors més usuals per aquesta n que anomenem mini-batch oscil · len entre 50 i 250 Leung and Haykin (1991)

Els algoritmes mencionats anteriorment no s'utilitzen explícitament durant el treball però sí els que es detallen a continuació. Els següents mètodes van un pas més enllà i es construeixen a través de la intuïció aportada pels explicats anteriorment: treballen

d'una forma exhaustivament eficient reduint el nombre d'iteracions (*epochs*) que haurem de completar per tal assolir aquest mínim.

1.5.7.3 Algoritmes de segon ordre

El descens del gradient presenta grans dificultats quan tracta d'assolir el mínim global en una funció multivariant on algunes regions presenten grans variacions en el pendent. Per tal d'esquivar aquest problema es van desenvolupar un conjunt d'algoritmes que utilitzen tasses d'aprenentatge adaptatives.
Aquest mètode és conegut com a *momentum* i es basa en l'actualització dels pesos no únicament del gradient calculat en la iteració i sinó a través de gradients calculats en interaccions anteriors.

1.5.7.3.1 ***AdaDelta*** AdaDelta parteix de l'algoritme AdaGrad, que en comptes de treballar amb una tassa d'aprenentatge fixa com els mètodes vists fins al moment, modifica la tassa d'aprenentatge η a cada període t.
Utilitzem la notació $g_{t,i}$ per la derivada parcial de la funció objectiu respecte del conjunt de paràmetres $Ttheta$ a l'instant t, és a dir:

$$g_{t,i} = \nabla_{\Theta} C\left(\Theta_{t,i}\right) \tag{1.5.9}$$

AdaDelta en comptes d'utilitzar el vector $g_{t,i}$ sencer, fixa una finestra de mida w i utilitza la mitjana mòbil exponencial en comptes de la suma de tots els gradients:

$$\mathbb{E}\left[g^2\right]_t = \gamma \mathbb{E}\left[g^2\right]_{t-1} + (1-\gamma) g_t^2 \tag{1.5.10}$$

A més, AdaDelta, tracta de solucionar el principal inconvenient que presenta AdaGrad que no és més que el ràpid decaïment del valor η cap a zero.
Els pesos s'actualitzaran utilitzant la següent expressió:

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{\mathbb{E}\left[g^2\right]_t + \epsilon}} \cdot g_t \tag{1.5.11}$$

1.5.7.3.2 ***Rsmprop*** Rmsprop és un algoritme que es va desenvolupar al mateix temps que AdaDelta per tal de solucionar el problema mencionat anteriorment respecte al ràpid

decaïment de la tassa d'aprenentatge adaptativa a zero.

Rmsprop és idèntic a AdaDelta però fixant el valor del paràmetre $\gamma = 0.9$

$$\begin{aligned} E\left[g^2\right]_t &= 0.9E\left[g^2\right]_{t-1} + 0.1g_t^2 \\ \Theta_{t+1} &= \Theta_t - \frac{\eta}{\sqrt{E\left[g^2\right]_t + \epsilon}} g_t \end{aligned} \tag{1.5.12}$$

1.5.7.3.3 ***Adam*** Un dels algoritmes més emprats en l'actualitat per la seva eficiència i robustesa. Adam utilitza el concepte de momentum i actualitza els pesos basant-se no únicament amb el gradient actual sinó afegint fraccions de gradients anteriors. Ho fa d'una forma adaptativa assignant un coeficient diferent per cada paràmetre.

Adam, en comptes d'utilitzar una mitjana mòbil exponencial com el cas d'AdaDelta, guarda la mitjana i variància del vector de gradients i afegeix uns termes β_1 i β_2 per tal d'evitar biaixos quan el vector conté valors propers a zero.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1)\, g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2)\, g_t^2 \end{aligned} \tag{1.5.13}$$

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \end{aligned} \tag{1.5.14}$$

Per últim actualitza els pesos de la següent forma:

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{1.5.15}$$

1.5.7.4 Tipus de xarxes neuronals

En els apartats anteriors es definien alguns conceptes bàsics sobre xarxes neuronals i els processos necessaris per a dotar aquestes xarxes amb la capacitat d'aprendre. Aquesta secció intenta mostra una possible classificació de les xarxes neuronals en funció de la connexió entre capes i la forma que propaguem els valors de sortida una neurona.

Xarxes *feed-forward*: En aquest tipus de xarxes el valor de sortida d'una neurona és sempre propagat cap endavant; els valors de sortida de les neurones no influeix de cap

forma el valor d'activació de les capes anteriors.

Xarxes neuronals recurrents: Són xarxes que es basen en la retroalimentació entre la capa de sortida i la capa d'entrada aportant dinamisme a la xarxa. Algunes d'aquestes xarxes utilitzen aquest comportament dinàmic fins a un cert punt on s'estabilitzen i existeixen altres on el valor de sortida influeix sempre sobre la xarxa. La majoria d'aquestes xarxes tenen una precisió més baixa que les convencionals tot i que són una arquitectura molt interesant, ja que s'assemblen molt més al comportament del cervell humà.

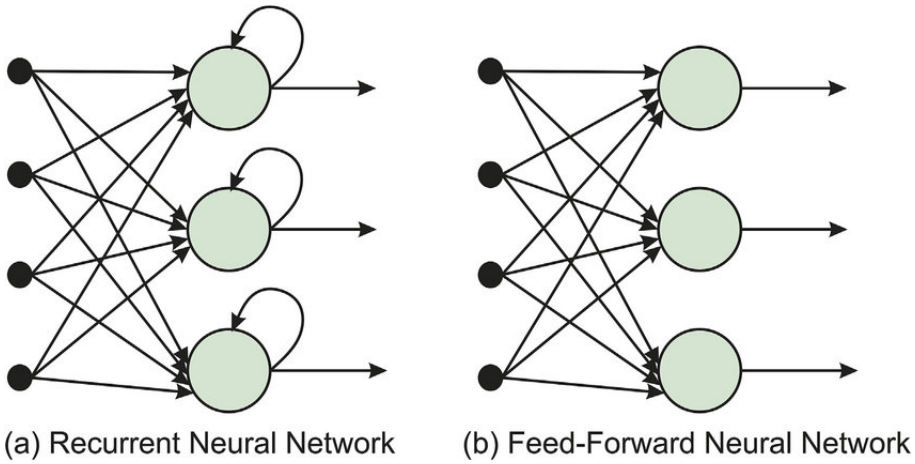


Figura 1.5: Tipus de xarxes neuronals segons la connexió entre capes.
(Font: Elsevier 2020)

1.6 L'autoencoder

En aquesta secció es mostra una de les estructures més senzilles que es poden crear a través d'una xarxa neuronal, l'autoencoder. Es tracta d'un model d'aprenentatge no supervisat que tracta de reconstruir les dades inicials després de comprimir-les en una dimensió menor.

Aquesta arquitectura profunda es basa dos elements principals: l'*encoder* i el *decoder*. El primer element s'encarrega de la compressió de les dades en un espai dimensional menor que l'espai original de les dades i el segon element rep aquesta compressió i reconstrueix novament les dades a l'espai original.

L'objectiu d'aquest model és la reducció de la dimensionalitat a través d'aquesta compressió. Aquesta compressió ha de ser efectiva per tal de mantenir l'estructura de les dades un cop reconstruïdes.A més, és interessant observar en detall aquesta compressió ja que

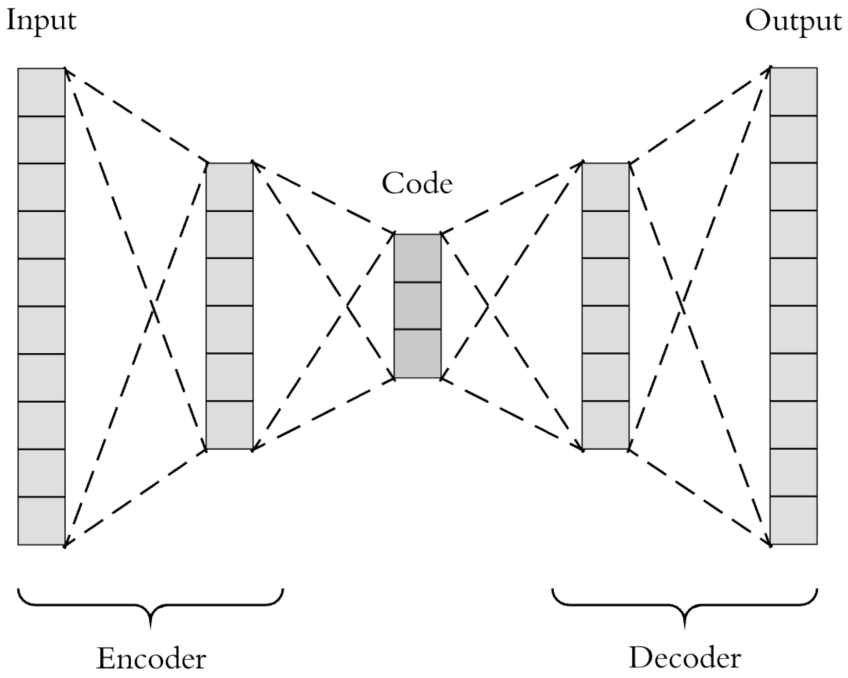


Figura 1.6: Estructura d'un autoencoder bàsic.
(Font: Gad 2020)

sovint serem capaços d'extreure informació rellevant sobre la distribució de les nostres dades quan es projecten en una dimensió menor de forma similar quan ho fem a través de les PCA.

1.7 Xarxes neuronals convolucionals

Les xarxes neuronals convolucionals (CNN) són un altre tipus de xarxa que sorgeixen als anys 80 creada a partir de la replicació del funcionament còrtex visual del cervell. Han demostrat molt bons resultats en la classificació d'imatges o la conducció automàtica, sense limitar-se a tasques relacionades amb la percepció visual: també són àmpliament utilitzades en altres tasques, com ara el reconeixement de veu o el processament de llenguatges naturals (NLP).
Existeixen diferents tipus de xarxes convolucionals però daurant aquest projecte s'utilitzaran les més populars: les *ConvCNN*. Es tracten de xarxes *feed-forward* que a diferència del perceptró multicapa on les dades d'entrada es presenten en un vector, aquesta xarxa admet objectes com imatges de tres dimensions (llargària, amplitud i nombre de canals).
Les CNN utilitzen tres operacions bàsiques per tal d'assolir els objectius mencionats anteriorment:

- Convolució
  L'objectiu principal d'aquest pas és extreure les característiques que permeten

identificar de la imatge d'entrada. A través d'un producte escalar entre la imatge i una matriu que rep el nom de *kernel* la convolució crea unes representacions de les imatges anomenades com mapes de característiques. Aquests conserven l'estructura principal de les imatges i a través de la utilització de més d'un *kernel* per tal de recollir les relacions menys superficials podem generar una bona representació matemàtica.

- Introducció de la no-linearitat
  També anomenat com *Rectified Linear Unit* (ReLU), és una operació extra de la convolució que té per objectiu reemplaçar els píxels negatius per zero per tal d'introduir la no-linearitat que la majoria de relacions entre els píxels d'una imatge complexa contenen.

- *Pooling* Un cop ha finalitzat el procés de la convolució, reduïm la dimensionalitat dels nostres mapes de característiques a la vegada que retenim la informació més important d'aquests. Existeixen diferents tipus de *pooling*, sent *Max Pooling* un dels més utilitzats en l'actualitat.

La següent figura representa el procediment explicat. També s'anima al lector a revisar el recurs Serrano (2017) si vol aprofundir en el tema.

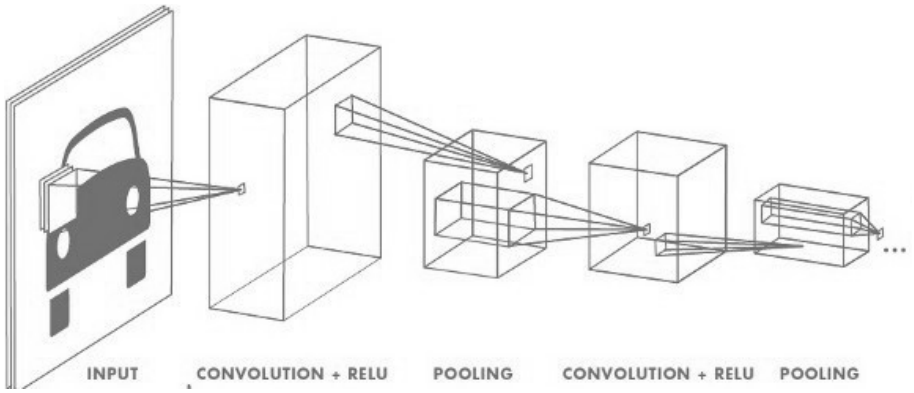


Figura 1.7: Algoritme principal basat en CNNs. La imatge mostra com el *input* passa per les fases descrites anteriorment.

(Font: Chaudhary 2020)

Capítol II

Quantificar la informació

"We are drowning in information but starved for knowledge" - John Naisbitt, "Megatrends" (1982)

2.1 Què entem per informació?

La informació és un dels béns crucials del segle XXI. Les dues darreres dècades hem experimentat un augment espectacular de la quantitat de informació i dades que s'emmagatzemen en format electrònic. Aquesta acumulació de dades s'ha produït a un ritme explosiu. S'ha estimat que la quantitat d'informació al món es duplica cada 20 mesos i que la mida i el nombre de bases de dades augmenten encara més ràpidament.

És fàcil confondre els conceptes de dades i informació. Encara que estan fortament lligats no podem assumir que són iguals. Un conjunt de dades pot ser enormement extens i no contenir cap mena d'informació rellevant. Plantegem-nos per un moment com podem quantificar la informació: ens adonarem de seguida que no és un problema trivial i que sembla impossible discernir si un conjunt de dades és o no informatiu *a priori* sense aplicar cap tècnica estadística.

Motivat per aquesta preocupació, Claude Shannon va delimitar el concepte d'informació a partir de l'expressió següent:

$$\mathrm{I}(x) := -\log_b \left[\Pr\left(x\right)\right] = -\log_b \left(P\right) \tag{2.1.1}$$

Lligant el concepte de probabilitat i informació, Shannon va aconseguir una mesura fent que sigui més senzill quantificar aquesta. A partir d'aquesta definició es construirà el camp de la teoria de la informació, que deixarà empenta en el món de la IA. Aquesta

definició queda recollida a Ash (2010).

Aquesta definició d'informació compleix els següents axiomes:

Axioma 1 *Un esdeveniment amb probabilitat 1 és completament previsible i no aporta cap mena d'informació.*

Axioma 2 *Com més improbable és un esdeveniment, menys previsible és i més informació ens aporta.*

Axioma 3 *Si es quantifica per separat la informació de dos esdeveniments independents, la quantitat total d'informació és la suma de les informacions individuals de cadascun dels esdeveniments.*

2.2 L'entropia de Shannon

$$H(X) = -\sum_i P_X(x_i) \log_b P_X(x_i) = \sum_i P_X(x_i) I_X(x_i) = \mathrm{E}[I_X] \tag{2.2.1}$$

Podem entendre l'anterior expressió com una forma de mesurar la quantitat d'informació esperada en recollir una certa variable X.

2.3 La divergència Kullback–Leibler

Un concepte fortament lligat a l'entropia és la divergència Kullback–Leibler. Podem pensar en aquesta mesura com una forma de quantificar com el nostre esdeveniment A diferent de B des de la perspectiva de A. Definim aquesta mesura per tal de facilitar la comprensió:

Definició 1 (Divergència Kullback–Leibler (sobre dos esdeveniments)) *Siguin A i B dos esdeveniments definits sobre el mateix espai de probabilitat $\mathcal{X}$, definim la divergència Kullback–Leibler com:*

$$D_{KL}(A \parallel B) = \sum_i P_A(x_i) \log P_A(x_i) - P_A(x_i) \log P_B(x_i) = \sum_i P_A(x_i) \log \left(\frac{P_A(x_i)}{P_B(x_i)} \right) \tag{2.3.1}$$

On P_A i P_B representen la probabilitat d'ocurrència dels esdeveniments A i B respectivament.

Nosaltres, però, ens centrarem en la definició de la divergència KL sobre dues variables contínues, ja que durant tot el treball tractarem majoritàriament amb distribucions de probabilitat d'aquest tipus. Podem entendre ara aquesta divergència com una mesura de

la diferència d'una distribució Q determinada respecte a una altra distribució P escollida com la distribució de referència.

Definició 2 (Divergència Kullback–Leibler (cas continu)) *Siguin P i Q dues distribucions contínues definides sobre el mateix espai de probabilitat $\mathcal{X}$, definim la divergència Kullback–Leibler com:*

$$D_{KL}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx \tag{2.3.2}$$

on p i q representen les funcions de densitat de P i Q respectivament

Tres propietats recollides en Bishop (2006) que convé mencionar sobre aquesta mesura són:

1. La divergència KL és sempre positiva. $D_{\mathrm{KL}}(P \parallel Q) \geq 0$,

2. Utilitzant la desigualtat de Gibbs (7.4), la divergència KL val zero si i només si $P = Q$.

3. Encara que aquesta mesura sovint s'entén com una mesura de distància entre dues distribucions, això no és cert des d'un punt de vista rigorós. La divergència KL no és una mètrica donat que no és simètrica i no satisfà la desigualtat triangular (7.5). Podem dir en general que $D_{\mathrm{KL}}(P \parallel Q) \neq D_{\mathrm{KL}}(Q \parallel P)$

Tots aquests conceptes introduïts prèviament seran clau per tal de definir diferents elements que formen el nostre VAE i és imprescindible conèixer-los.

Capítol III

Modelització probabilística a través de xarxes neuronals

3.1 Introducció als models probabilístics a través de xarxes neuronals

A continuació es tracta d'explicar el plantejament que s'acostuma a seguir quan es parla de les xarxes neuronals com a models probabilístics. Imaginem que estem davant d'un problema de classificació binària, on tractem de classificar una imatge en diferents categories d'acord amb una etiqueta que és coneguda en cada exemple. Podem entendre una imatge com un punt en un espai de milers o milions de dimensions, on en cada una d'aquestes dimensions està representada un píxel.

En aquest punt sabem que existeix una distribució teòrica perfecta sobre les nostres dades que a assigna probabilitat 1 a la categoria corresponent i 0 a la resta. Aquesta distribució però, és desconeguda i molts cops molt difícil d'estimar. A través de mètodes estadístics clàssics com la regressió logística podríem tractar d'obtenir una distribució aproximada encarregada d'assignar una probabilitat a cada una de les classes per cada imatge.

Podríem fer ús d'alguna funció per tal de quantificar l'error que cometem en classificar aquesta imatge i intentar minimitzar aquest error a través de l'ajust d'uns paràmetres que ens permetin minimitzar la funció de pèrdua que hàgem escollit. Justament és aquesta la idea darrere de les xarxes neuronals. Podem entrenar una xarxa de forma que sigui capaç d'aprendre a classificar aquestes imatges a través de l'ajust dels paràmetres corresponents.

Un cop explicada la intuïció, tractem de formalitzar l'anterior:

Assumim ara que la nostra variable observada x és una mostra aleatòria d'una distribució desconeguda $p^*(x)$. El nostre objectiu serà aproximar aquesta distribució real a través d'un model $p_\theta(x)$ amb paràmetres θ que optimitzarem a través d'algun mètode basat en el descens del gradient. Aquest model, de fet, és possible que no sigui un model estadístic conegut, sinó simplement una funció $f_\theta(x)$ de forma que $f_\theta : x \mapsto [0, 1]$. Aquesta funció, a la pràctica, vindrà modelada per una xarxa neuronal que utilitzant la funció sigmoide ens retornarà valors entre 0 i 1.

$$x \sim p_0(x) \tag{3.1.1}$$

Direm que el nostre model ha après la distribució real de les dades si és capaç de trobar un valor pel paràmetre θ de forma que el model escollit descriu la distribució de les dades de forma aproximada. És a dir, hem trobat un model p_θ tal que:

$$p_0(x) \approx p^*(x) \tag{3.1.2}$$

Tractarem de definir $p_0(x)$ de la forma més flexible possible per tal que el nostre model s'adapti a les nostres dades de forma general.

3.1.1 Models probabilistics condicionals

Quan tenim informació a priori sobre les nostres dades, és interessat introduir aquesta per tal de millorar els nostres resultats. És aquí on entren els models condicionals. De fet, quan estem davant d'un problema de classificació o de regressió, no busquem estimar un model de la forma $p_0(x)$ sinó una model condicional $p_0(y|x)$ per tal d'aproximar la nostra distribució condicionada real. Formalitzant aquest raonament, podem dir que busquem $p_0(y|x)$ de forma que:

$$p_0(y|x) \approx p^*(y|x) \tag{3.1.3}$$

Un exemple clar d'això és la classificació d'imatges. Suposem ara que x és una

imatge i y és la categoria a la qual pertany aquesta imatge. En aquest tipus de problemes buscarem modelitzar $p(y|x)$ com una distribució categòrica que assigni un valor a cadascuna de les categories d'acord amb les característiques de x.

En aquest cas, podriem dir que busquem:

$$p_\theta(categoria \mid dades) = \frac{p_\theta(\text{ dades } \mid \text{ categoria })p_\theta(\text{ categoria })}{p_\theta(\text{ dades })} \tag{3.1.4}$$

O de forma més formal:

$$p_\theta(y \mid x) = \frac{p_\theta(\text{ x } \mid \text{ y })p_\theta(\text{ y })}{p_\theta(\text{ x })} \tag{3.1.5}$$

Els paràmetres θ s'esculliran d'acord amb, és a dir:

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} p_\theta\left(y^{(i)} \mid x^{(i)}\right) \tag{3.1.6}$$

Encara que molts cops s'inclou el logaritme dins d'aquesta funció per tal d'evitar problemes de càlcul

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} log\ p_\theta\left(y^{(i)} \mid x^{(i)}\right) \tag{3.1.7}$$

Encara que pugui arribar a semblar una notació una mica forçada, és convenient introduir aquests termes per tal que el lector no és sorprengui quan s'apliquin tècniques inferencials i es tractin els models probabilístics neuronals com a models estàndard en segons quins moments.

3.1.2 Intractabilitat

Un dels grans problemes en el món dels models en grafs és la coneguda intractabilitat de la probabilitat marginal $p_\theta(x)$. Aquesta probabilitat és calculada com $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{y})p_\theta(y)d\mathbf{y}$ i és intractable des de el punt de vista computacional donat l'alt cost de calcular aquesta integral en espais R^n on n és molt gran com acostuma a ocórrer quan es treballa amb imatges. Arribats a aquest punt, existeixen dos enfocaments

que busquen solucionar aquesta problemàtica.

1. Mètodes de Monte Carlo: Els mètodes de Monte Carlo busquen trobar el valor de $p_\theta(x)$ a través de la simulació.

2. Infèrencia variacional: És el mètode en el qual es basa l'autoencoder variacional i que es tracta en aquest treball. S'observarà en detall més endavant.

Podem pensar en els dos mètodes anterior com un *trade-off* entre variància i biax: el mètode de Monte Carlo que aproxima la integral a través del mostreig, es tracta d'un mètode no esbiaixat amb alta variància. Per contra, el segon mètode que busca un model que aproximi a la distribució és un mètode esbiaixat amb una baixa variància.

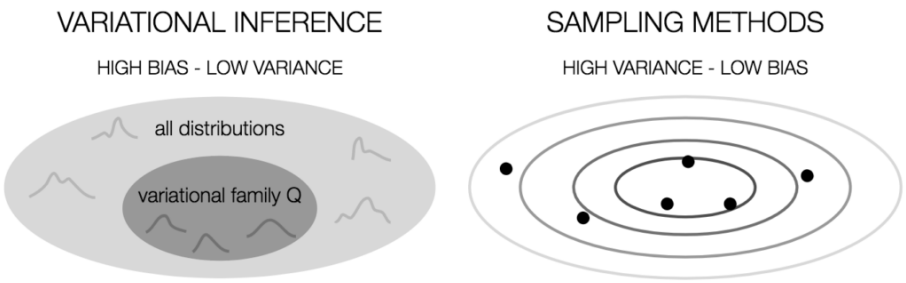


Figura 3.1: Representació gràfica del *trade-off* entre els mètodes de Monte Carlo i la infèrencia variacional

(Font: Zeldes 2018)

3.2 Els models generatius: una visió general

Els models generatius són una extensa àrea d'aplicació de l'aprenentatge no supervisat dins del Machine Learning. Aquests models parteixen d'un conjunt de dades d'entrada i tenen com a objectiu principal la generació de noves dades.

Tots els models generatius tenen com necessiten aprendre la distribució real de les dades per tal d'assolir el seu objectiu. Es busca per tant una distribució de probabilitat $p_\theta(y \mid x)$ o $p_\theta(x)$ en cas de no tenir etiquetes de no disposar d'etiquetes de classes per tal de discernir patrons que comparteixin les nostres dades.

Aprendre aquesta distribució vol dir ser capaç de generar noves mostres seguint la mateixa estructura d'aquelles que es troben a la base de dades. Aquesta tasca però, és una de les més complicades dins del món del Machine Learning. En la majoria dels casos aquesta distribució és impossible de determinar i es tracta d'aproximar-la de la forma més exacta possible.

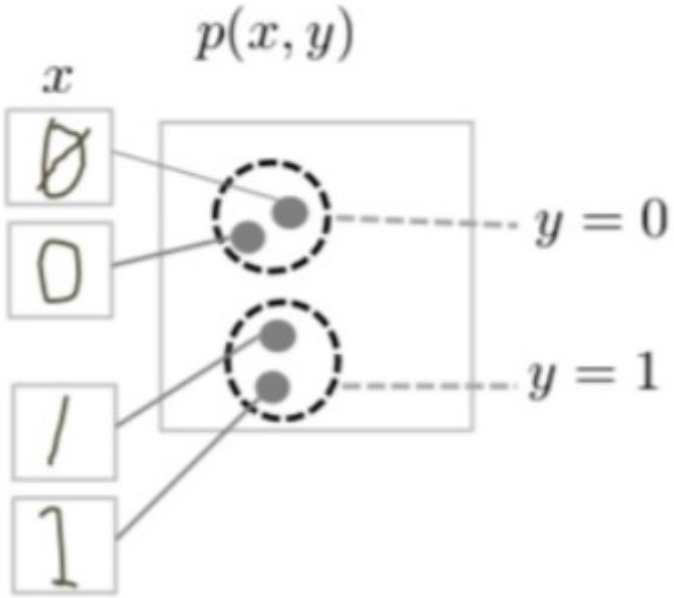


Figura 3.2: Procès de esquemàtic dels models generatius.
(Font: Goodfellow 2016)

3.2.1 Estructura dels models generatius

Les aplicacions dels models generatius no es limiten VAE estudiat. De fet, els models generatius siguin uns dels temes més interessants dins dels models d'aprenentatge profund en l'actualitat i gran part de la comunitat científica sembla estar realment interessada en les aplicacions d'aquests models. A continuació es mostra la classificació més acceptada d'aquest tipus de models.

3.2.2 Models de densitat explícita

Aquests models tracten amb la funció de densitat explícita de les dades i ho fan des de dos punts de vista totalment diferents: uns busquen de maximitzar directament una densitat explícita $p_\theta(x)$ i altres aproximant aquesta distribució a través de funcions similars a $p_\theta(x)$ de forma que es busca $q_\theta(x)$ tal que $q_\theta(x) \approx p_\theta(x)$.

Tot i que sembla que alguns dels primers mètodes com pixelRNN estan donant bons resultats són molt costosos computacionalment. Estimar aquesta funció sembla bastant raonable i emprem els mètodes mencionats en la secció 2.1 - Monte Carlo i inferència variacional - per tal de solucionar la coneguda intractabilitat de la integral de p(x) En aquesta segona subdivisió trobem el VAE, el model estudiarem en profunditat durant tot el treball.

3.2.3 Models de densitat implicita

Aquests models no tracten d'estimar cap classe de funció de densitat probabilística sinó que tracten el problema d'una forma directa i empírica, tracten de discernir entre dades creades de forma artificial i dades reals per tal d'aprendre la distribució latent de les dades. L'exemple més conegut és el GAN que s'explica en detall a continuació.

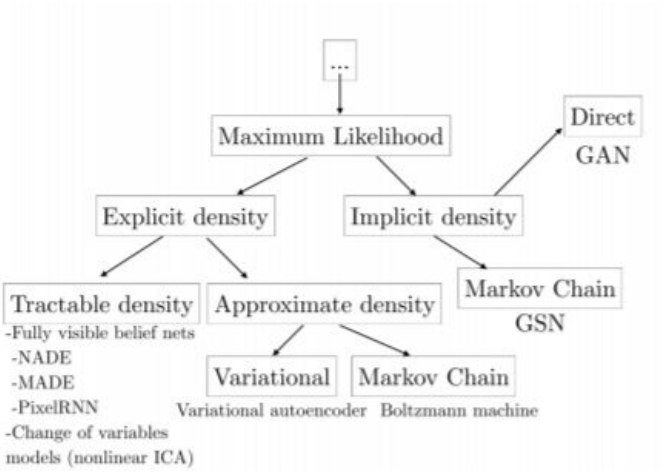


Figura 3.3: Taxonomia dels models generatius.
(Font: Goodfellow 2016)

És interessant observar un altre enfocament al problema que tractarem sobre la generació de noves dades. Els models GANs és l'alternativa clàssica als autoencoders variacionals. Es tracta d'un dels principals models generatius de dades que s'utilitzen actualment, basat principalment en l'entrenament de dos models d'aprenentatge profund per separat. Es pot veure més en detall en 7.9.

Part I

L'autoencoder variacional

Capítol IV

Fonaments teòrics

4.1 Introducció

Els *autoencoders* variacionals han sigut tradicionalment, com ja s'ha explicat, un model de compressió que ha fet possible la reducció de la dimensionalitat a través de representació de les variables en un espai dimensional més petit. La projecció a aquest espai latent ens aporta una codificació molt interessant de les dades, donat que podem extreure patrons i relacions que recullen l'estructura de les dades i són de gran utilitat quan reconstruïm les imatges.

Podem identificar els elements del nostre VAE amb un *autoencoder* estàndard, podem pensar llavors que l'element que comprimeix les nostres dades en l'*autoencoder*, és a dir, aquell que donat un element del data set ho codifica en un espai de menor dimensió, serà ara el nostre model de reconeixement $Q_\phi(z \mid x)$, encarregat ara de realitzar aquesta compressió de forma que podem extreure l'estructura i els patrons de les nostres dades. L'espai de codificació de l'*autoencoder* tradicional ho anomenarem ara com l'espai latent i contindrà les variables z. Aquest haurà de complir uns certs requisits que no eren necessaris en l'*autoencoder* tradicional per tal de ser un bon model generatiu. Per últim, l'element que transforma l'espai latent en una imatge reconstruïda podria ser pensat com un *decoder* probabilístic $P_\theta(x \mid z)$

Per tal de modelitzar $Q_\phi(z \mid x)$ i $P_\theta(x \mid z)$ podem utilitzar qualsevol classe de xarxa neuronal. L'elecció més freqüent però acostuma a ser un perceptró multicapa. En el cas de modelitzar relacions entre píxels d'imatges s'acostuma a emprar les xarxes convolucionals. Quan utilitzem aquest últim tipus de capes (1.7) parlem d'un *autoencoder* variacional convolucional **(Conv-VAE)**.

Esquemàticament, el procés del VAE seria:

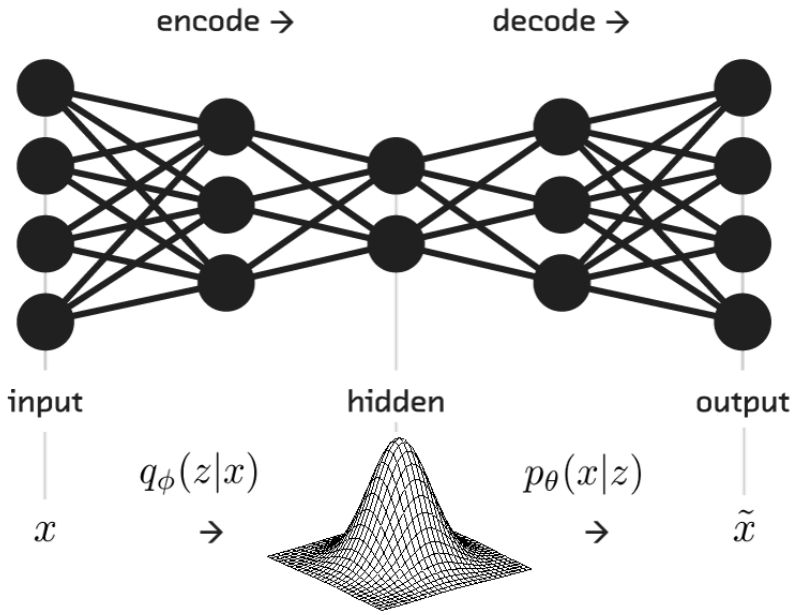


Figura 4.1: Elements que conformen el VAE
(Font: Kamruzzaman and Tappert 2019)

- Rep un conjunt de dades $X = \{x_1, \ldots, x_N\}$ que consisteix en N mostres i.d.d. sobre algun espai dimensional $\mathbb{R}^N$
- Crea l'espai lantent que conté les variables $z = \{x_1, \ldots, x_N\}$ utilitzant el model de reconeixement $Q_\phi(x \mid z)$ i a partir d'aquesta distribució mostreja una z aleatòria.
- Aquesta variable latent z que és generada a través d'una distribució $P(z)$ és introduïda ara al decoder probabilístic, que s'encarregarà de reconstruir la imatge.

La flexibilitat del VAE fa que pugui ser àmpliament utilitzat en situacions diverses. Donades les característiques mencionades de l'espai latent, les aplicacions del VAE passen fins a la reducció de la dimensionalitat, la generació d'una codificació útil emprada en una possible futura modelització de les dades o fins a la més clàssica de totes; la generació de noves dades. Durant aquest projecte seguirem aquesta última línia i tractarem de generar noves imatges a partir d'un conjunt de dades.

4.2 Definició de la funció objectiu

El VAE, com a model generatiu de densitat explícita tracta de maximitzar la funció de densitat $p(x)$. Aquesta, com ja s'ha mencionat anteriorment, es pot reescriure com:

$$P(\boldsymbol{x}) = \int P_\theta(\boldsymbol{x} \mid z)P(z)dz \tag{4.2.1}$$

Introduïm el logaritme per millorar l'estabilitat numèrica i reescrivim la funció per tot el conjunt de dades $\{x_1, \ldots, x_N\}$

$$\log P_\theta\left(x_1, \ldots, x_N\right) = \sum_{i=1}^{N} \log P_\theta\left(\boldsymbol{x}_i\right) = \int P_\theta(\boldsymbol{x_i} \mid z)P(z)dz \tag{4.2.2}$$

Observem l'equació 4.2.1 amb més detall. En cas que poguéssim conèixer tot l'espai on està definit z, si decidim aproximar $\log P_\theta$ a través d'un mostreig complet, la majoria dels termes $P_\theta(x \mid z)$ seran propers a zero i no contribuiran en la maximització de la funció de densitat.

Cal mencionar que aquest problema estaria resolt directament si fóssim capaços de calcular analíticament la probabilitat $P(z \mid x)$:

$$P(z \mid x) = \frac{P(x \mid z)P(z)}{P(x)} \tag{4.2.3}$$

Però aquí ens apareix de nou el terme $P(x)$ que com ja sabem es dóna la coneguda intractabilitat. La pregunta que apareix de forma natural llavors és per tant com definir les variables z.

És en aquest punt on entra en joc el nostre model de reconeixement $Q_\phi(z|x)$ (o compressor si ho pensem des del punt de vista d'un *autoencoder* tradicional). Ser capaç de recrear un espai latent z implica que els valor de $P(X \mid z)$ estaran més lluny de 0 que si escollim una z arbitraria. Si som capaços de trobar una distribució de probabilitat $Q(z \mid x)$ de forma que els valor de z siguin informatius i útils per recrear z estarem fent així el nostre VAE més eficient. Escollirem $Q_\phi(z|x)$ per tant de forma que sigui el més similar possible a $P(z \mid x)$.

A partir d'aquest moment haurem de fer ús de l'estadística inferencial per tal d'inferir la distribució $P(X \mid z)$ a través de l'aproximació $Q_\phi(z|x)$.

La mesura que utilitzarem per quantificar les diferències entre dues distribucions serà la divergència KL. Aquesta ja s'ha introduït en l'apartat de conceptes previs i s'ha presentat amb més detall.

$$\mathcal{D}[Q(z \mid x) \| P(z \mid X)] = E_{z\sim Q}[\log Q(z) - \log P(z \mid X)] \tag{4.2.4}$$

Introduïm en l'equació $P(X \mid z)$ i $P(X)$ a través del teorema de Bayes tal com s'observa en

$$\mathcal{D}[Q(z) \| P(z \mid X)] = E_{z\sim Q}[\log Q(z) - \log P(X \mid z) - \log P(z)] + \log P(X) \tag{4.2.5}$$

Reescrivim els termes de l'equació i finalment obtenim:

$$\log P(X) - \mathcal{D}[Q(z \mid X) \| P(z \mid X)] = E_{z\sim Q}[\log P(X \mid z)] - \mathcal{D}[Q(z \mid X) \| P(z)] \tag{4.2.6}$$

És freqüent però trobar aquesta equació expressada de la següent forma Hoffman et al. (2013):

$$\log P(x) = ELBO(\phi, \theta) + \mathcal{D}_{KL}\left(Q_\phi(z \mid x) || P_\theta(z \mid x)\right) \tag{4.2.7}$$

On $ELBO(\phi, \theta)$:

$$ELBO(\phi, \theta) = E_{z\sim Q_\phi}\left[-\log Q_\phi(z \mid x) + \log P_\theta(x, z)\right] = \tag{4.2.8}$$

$$-\mathcal{D}_{KL}\left(Q_\Phi\left(z \mid x\right) \| P_\theta(z)\right) + E_{z\sim Q_\phi}\left[\log P_\theta\left(x \mid z\right)\right] \tag{4.2.9}$$

Normalment la idea d'expressar l'anterior equació d'aquesta forma és pensar que el fet de maximitzar l'$ELBO(\phi, \theta)$ implica minimitzar la discrepància entre les distribucions $-\mathcal{D}_{KL}\left(Q_\Phi\left(z \mid x\right) \| P_\theta(z)\right)$ i millorar de forma directa el valor de $\log P(x)$.

4.3 Elecció de les distribucions $Q_\phi(z|x)$

Anteriorment hem definit la nostra funció objectiu per qualsevol distribució probabilística. Per tal de portar el nostre model a la pràctica, haurem de prendre una decisió respecte a quina distribució escollirem pel nostre espai latent així com pel *decoder* probabilístic.

El terme $P(z)$ definirà el nostre espai latent i la distribució de les nostres variables z. Per tal de generar un espai homogeni on es pugui projectar les característiques de les nostres dades que seran utilitzades després per generar noves mostres, els autors del VAE utilitzen la distribució $N(0, I)$. Aquesta distribució ens assegura que el nostre espai latent serà continu i no presentarà discontinuïtats.

Respecte al *decoder*, és molt freqüent l'elecció $N(\mu(X), \Sigma(X))$ per motius computacionals. El fet d'escollir una distribució normal pel nostre fa que podem computar la divergència de dues distribucions multivariants gaussianes de forma tancada:

$$\mathcal{D}[\mathcal{N}(\mu(X), \Sigma(X)) \| \mathcal{N}(0, I)] = \tag{4.3.1}$$

$$\frac{1}{2}\Big(\mathrm{tr}(\Sigma(X)) + (\mu(X))^\top(\mu(X)) - k - \log\det(\Sigma(X))\Big) \tag{4.3.2}$$

4.4 Optimització de la funció objectiu

Podem dividir la nostra funció objectiu 4.2.6 en dos grans blocs. El terme $E_{z\sim Q}[\log P(X \mid z)]$ indica l'error de reconstrucció de la imatge. Aquest terme és àmpliament utilitat en els diferents models de compressió i denota la precisió amb la qual el VAE pot reconstruir la imatge original.

El terme anterior busca que l'encoder creï un espai latent representatiu de la imatge d'entrada, de manera que basant-se en aquest espai de dimensió reduïda, el *decoder* assigni una alta probabilitat a la imatge original. La idea darrere d'aquest terme és que si suposem que els nostres models treballen amb els paràmetres ϕ, θ òptims, la representació latent ha de ser necessàriament informativa per tal que el terme $P_\theta(x \mid z)$ prengui un valor proper a 1.

A la pràctica quantificarem aquest terme utilitzant l'entropia binària, una mesura que tot i que sembla que els resultats poden ser millorables amb l'ús d'altres mesures més avançades Huszár (2015) és l'elecció més usual.

Respecte als termes que utilitzen la divergència KL per quantificar similituds es poden

entendre de la següent forma:

- El primer terme $\mathcal{D}[Q(z \mid X) \| P(z \mid X)]$ és una clara penalització per no utilitzar el millor *encoding*. Aquest terme mesura la pèrdua d'informació quan estimem $P(z|x)$ a través de $Q_\phi(z|x)$.

- Respecte al segon terme $\mathcal{D}[Q(z \mid X) \| P(z)]$ situat al final de l'equació millora la robustesa del nostre model.
  Un cop observat un nou individu del nostre conjunt de dades, és possible que hàgim d'ajustar els paràmetres ϕ, θ del nostre model per tal d'adaptar-se a aquesta nova observació d'una forma massa brusca si es tractés d'un possible *outlier*. Lligant el nostre decoder probabilístic i l'espai latent a través d'aquesta mesura de discrepància evitem que limitant aquest efecte fent que el nostre decoder no perdi de vista les característiques de totes les altres observacions que s'han recollit en aquest espai latent $P(z)$.

Un cop hem entès la intuïció darrera de la nostra funció objectiu, només ens cal aplicar un algoritme d'optimització que aconsegueixi maximitzar la nostra funció objectiu a través de l'ajust dels paràmetres ϕ, θ.

Ràpidament detectarem la dificultat que se'ns presenta: en el moment de propagar l'error cap enrere, observarem que una de les capes pren mostres de $q_\theta(z|x)$. Detectem llavors que aquesta operació de mostreig no és diferenciable i per tal d'aplicar el descens del gradient (o qualsevol altre algoritme que busqui minimitzar la funció objectiu) haurem de complir aquest requisit.

4.5 La reparametrització

La solució a aquest problema es coneix com a reparametrització de la xarxa. En comptes de realitzar el mostreig directament d'una $N(\mu(X), \Sigma(X))$, prenem mostres d'una distribució normal del vector de mitjanes $\mu(X)$ sigui un vector de zeros i la matriu de variàncies i covariàncies $\Sigma(X) = I$. Un cop obtinguem aquesta mostra ho assignarem a una nova variable de forma que $e \sim N(0, 1)$ i calcularem z com $z = \mu(X) + \Sigma^{1/2}(X) * \epsilon$ movent així l'operació de mostreig a una capa d'entrada de la xarxa.

És mostra un esquema a continuació d'aquesta modificació:

La nostra funció objectiu serà ara:

$$E_{X \sim D}\left[E_{\epsilon \sim \mathcal{N}(0,I)}\left[\log P\left(X \mid z = \mu(X) + \Sigma^{1/2}(X) * \epsilon\right)\right] - \mathcal{D}[Q(z \mid X) \| P(z)]\right] \quad (4.5.1)$$

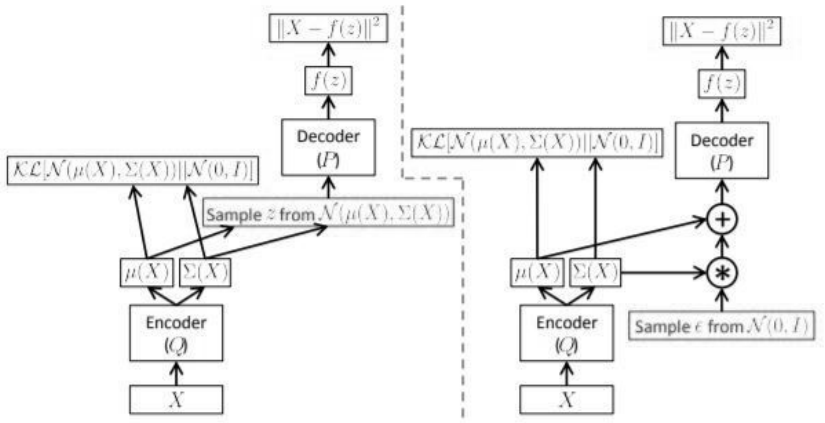


Figura 4.2: Reparametrització de la xarxa. En aquesta figura es mostra com reconfigurem la xarxa per tal d'extreure la operació no derivable fora.
(Font: Doersch 2016)

On hem aconseguit deslligar les esperances de la nostra funció objectiu de qualsevol paràmetre del model. Ara sí que podrem introduir l'operació del gradient dins de la nostra funció objectiu.

4.6 Generació de noves dades

Un cop el nostre model ha après l'estructura de les dades a través la maximització de la funció de densitat, podem generar noves mostres senzillament prenent mostres d'una distribució normal de forma que $z \sim N(0, I)$ i enviant aquestes al *decoder* tot seguit.

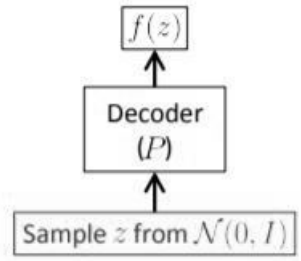


Figura 4.3: Procès de generació de noves mostres.
(Font: Doersch 2016)

4.7 L'òptica de la teoria de la informació

Donat que hem presentat prèviament alguns conceptes claus relacionats amb la teoria de la informació, és interessat entendre el nostre *autoencoder* variacional objectiu a través d'aquest prisma per tal d'aprofundir el coneixement sobre el nostre model. Analitzarem

per tant ala nostra funció objectiu que ens indica quins són els termes necessaris a tenir en compte per tal d'un correcte funcionament.

Observem la nostra funció objectiu per tant amb més detall. Podem pensar que log $P(X)$ és la quantitat d'informació necessària per a reconstruir el nostre conjunt de dades X utilitzant un espai latent ideal. Aquesta reconstrucció pot ser entesa a través dels dos termes costat dret de l'equació 4.2.6.

Respecte al primer terme, a l'introduir la suposició $P(z) = N(0, I)$ estem utilitzant una distribució que no ens aporta informació sobre l'espai latent. El terme $\mathcal{D}[Q(z \mid X) \| P(z)]$ llavors ens indica la contribució d'informació extra que la distribució $Q(z \mid X)$ ens dóna.

El segon terme és per tant la quantitat de bits que utilitzem per generar X utilitzant un espai latent idoni. Per tant, podem pensar que la quantitat de bits total requerida és la suma d'aquests dos termes, més una penalització $\mathcal{D}[P(z \mid X) \| Q(z \mid X)]$ que es dóna quan no utilitzem l'encoding ideal.

4.8 Problemes del VAE

4.8.1 Representativitat del espai latent

Un dels punts forts de l'autoencoder variacional és el seu espai latent. Donat que recull les característiques principals de les dades aquest és especialment útil per la generació de noves imatges entre altres tasques. Encara així, diversos investigadors han observat la possibilitat que aquest espai latent no sigui tan informatiu en algunes ocasions com es pensava.

El fet de lligar el nostre codificador amb la distribució latent de les dades a través de la divergència KL millora la robustesa del model, fent que l'ajust dels paràmetres sigui suau i més aconseguint que el nostre VAE sigui propens a comportaments més estables. Aquest terme però, presenta un inconvenient: el fet d'incentivar al nostre encoder a generar representacions latents similars als valors d'una distribució convenientment assumida anteriorment, dificulta la creació d'un espai latent realment informatiu, ja que estem condicionant d'alguna forma la configuració d'aquest espai.

4.8.2 Problemes inferencials

Tal com demostra Chen et al. (2016) a través del segon exemple el VAE presenta problemes en l'estimació del model $Q_\phi(z \mid x)$. Plantegem-nos ara següent situació: volem assolir l'òptim de la nostra funció objectiu en un dataset que conté dos punts {-1,1}

únicament, es pot demostrar llavors que utilitzant els encoders i decoders gaussians estàndards de forma que:

$$\begin{aligned} P_\theta(x \mid z) &= \begin{cases} \mathcal{N}(1, \sigma^2) & z \geq 0 \\ \mathcal{N}(-1, \sigma^2) & z < 0 \end{cases} \\ Q_\phi(z \mid x) &= \begin{cases} \mathcal{N}(c, \lambda^2) & x \geq 0 \\ \mathcal{N}(-c, \lambda^2) & x < 0 \end{cases} \end{aligned} \tag{4.8.1}$$

On $\sigma, c, \lambda \in \mathbb{R}$ i són els paràmetres que volem optimitzar, és possible maximitzar l'$ELBO(\phi, \theta)$ encara que $\lambda \to 0$, $c \to \infty$. Això demostra un comportament poc estable del VAE i poca representativitat del ELBO com a funció objectiu a maximitzar en alguns casos.

Per tal de provar l'anterior, maximitzarem els dos termes del ELBO suposant $\lambda \to 0$, $c \to \infty$ i observarem que ocorre.

El primer terme de l'ELBO quan x = 1 pot ser expressat com:

$$\begin{aligned} &E_{Q_\phi(z|x=1)}[\log P_\theta(x = 1 \mid z)] \\ &= Q_\phi(z \geq 0 \mid x = 1) \log P_\theta(x = 1 \mid z \geq 0) + \\ &Q_\phi(z < 0 \mid x = 1) \log P_\theta(x = 1 \mid z < 0) \\ &= Q_\phi(z \geq 0 \mid x = 1)(-1/2 \log(2\pi) - \log \sigma) + \\ &Q_\phi(z < 0 \mid x = 1)\,(-1/2 \log(2\pi) - \log \sigma - 2/\sigma^2) \\ &= -\log \sigma - 2Q_\phi(z < 0 \mid x = 1)/\sigma^2 - 1/2 \log(2\pi) \end{aligned} \tag{4.8.2}$$

Els paràmetres que ens permeten arribar a la solució òptima els podem trobar derivant l'expressió anterior respecte σ:

$$\frac{E_{Q_\phi(z|x=1)}}{\partial \sigma} = -1/\sigma + 4/\sigma^3 Q_\phi(z < 0 \mid x = 1) = 0 \tag{4.8.3}$$

On la igualtat anterior es dóna si i només si $\sigma = 2\sqrt{q(z < 0 \mid x = 1)}$

$$E_{Q_\phi(z|x=1)} = -1/2 \log Q_\phi(z < 0 \mid x = 1) + C \tag{4.8.4}$$

On C és una constant independent de c, λ i σ i $Q_\phi(z < 0 \mid x = 1)$ és la cua de probabilitat d'una distribució $N(\mu, \Sigma)$

L'expressió anterior, es pot provar Wei et al. (2020) que en el límit $\lambda \to 0$ i $c \to \infty$ es pot reescriure com:

$$E_{Q_\phi(z|x=1)} = \Theta\left(c^2/\lambda^2\right) \tag{4.8.5}$$

Per altra banda, respecte al segon terme del ELBO tenim:

$$-\mathcal{D}_{\mathrm{KL}}(Q_\phi(z \mid x = 1) \| P(z)) = \log \lambda - \lambda^2/2 - c^2/2 + 1/2 \tag{4.8.6}$$

I si unim els dos termes utilitzant l'operador límit, s'obté:

$$\begin{aligned} &\lim_{\lambda \to 0, c \to \infty} \mathcal{L}_{\mathrm{ELBO}}(x = 1) \\ &\qquad = E_{Q_\phi(z|x=1)} - \mathcal{D}_{\mathrm{KL}}(Q_\phi(z \mid x = 1) \| P(z)) \\ &\qquad = \Theta\left(c^2/\lambda^2 + \log \lambda - \lambda^2/2 - c^2/2\right) \\ &\qquad \to +\infty \end{aligned} \tag{4.8.7}$$

Llavors hem provat que podem maximitzar el $ELBO(\phi, \theta)$ en una situació d'inestabilitat.
A més, també es comprova que es falla a l'inferir la distribució de $P(z \mid X)$ a partir de $Q_\phi(z \mid X)$, ja que podem observar que quan $x = 1$ per qualsevol $z \in \mathbb{R}$:

$$\begin{aligned} P(z \mid x = 1) &= \frac{P_\theta(x = 1 \mid z)}{P(x = 1)} P(z) \\ &= \frac{P_\theta((x = 1 \mid z)}{\int_{z'} P_\theta\left(x = 1 \mid z'\right) P\left(z'\right) dz'} P(z) \\ &= \frac{P_\theta(x = 1 \mid z)}{1/2 P_\theta(x = 1 \mid z \geq 0) + 1/2 P_\theta(x = 1 \mid z < 0)} p(z) \\ &\leq \frac{2 P_\theta(x = 1 \mid z)}{P_\theta(x = 1 \mid z \geq 0)} P(z) \leq 2P(z) \end{aligned} \tag{4.8.8}$$

I això implica que la discrepància entre la distribució real de les dades i l'aproximada tendeix a infinit tal com s'observa a continuació:

$$\begin{aligned}
&\mathcal{D}_{\mathrm{KL}}(Q_\phi(z \mid x=1) \| P(z \mid x=1)) \\
&= E_{Q_\phi(z|x=1)}\left[\log \frac{Q_\phi(z \mid x=1)}{P(z \mid x=1)}\right] \\
&\geq E_{Q_\phi(z|x=1)}\left[\log \frac{Q_\phi(z \mid x=1)}{2P(z)}\right] \\
&= \mathcal{D}_{\mathrm{KL}}(Q_\phi(z \mid x=1) \| P(z)) - \log 2 \\
&\to \infty
\end{aligned} \tag{4.8.9}$$

Els resultats anteriors indiquen que VAE en comptes d'aconseguir que $\mathcal{D}_{\mathrm{KL}}(Q_\phi(z \mid x=1) \| P(z \mid x=1)) \to 0$ que és el que s'espera per tal d'aproximar la nostra distribució real, obtenim que $\mathcal{D}_{\mathrm{KL}}(Q_\phi(z \mid x=1) \| P(z \mid x=1)) \to \infty$.
Aquest comportament es tradueix que en comptes de generar un espai latent $P(z)$ homogeni aconseguim llavors dos models $\frac{Q_\phi(z|x=-1)}{2} + \frac{Q_\phi(z|x=1)}{2}$ separats l'un d'altre. Això és degut al fet que el terme aquest terme no està complint l'objectiu original sinó que divideix l'espai latent. Aquest comportament podria tenir sentit per tal d'evitar la superposició de classes en $P(z)$ però en comportar-se d'una forma límit, arruïna la inferència sobre el terme $P_\theta(z \mid x)$.

Capítol V

MMD-VAE

Per tal d'evitar els problemes que es mencionen anteriorment Wei et al. (2020) introdueixen una variació en la funció de pèrdua del nostre VAE que ens permetrà millorar l'estabilitat d'aquest i ser capaços de generar un espai latent més informatiu. Aquesta variació del VAE pren el nom de MMD-VAE (Discrepancia Màxima Mitjana VAE).

Estudiem ara que és aquest concepte amb més profunditat.

5.1 Discrepancia Màxima Mitjana

Imaginem ara que tenim $X := \{x_1, \ldots, x_n\} \sim p$ i $Y := \{y_1, \ldots, y_n\} \sim q$ i volem provar si $p = q$.

Comparar dues distribucions a través de la funció de densitat és un problema extensament estudiat i de difícil resolució pel conegut problema de la dimensionalitat. És per això que pot ser d'utilitat evitar computar la distribució directa de les dades i comparar els moments enlloc.

Amb aquesta idea (Fortet and Mourier, 1953) van presentar la següent mesura que té com objectiu comparar les discrepàncies de dues distribucions estimant les seves esperances, és a dir:

$$D(p, q, \mathcal{F}) := \sup_{t \in \mathcal{F}} \mathbf{E}_p[f(x)] - \mathbf{E}_q[f(y)] \tag{5.1.1}$$

Es pot provar (7.10)que el resultat de l'espai anterior quan es treballa en $\mathbb{R}^n$ és

$$E_{p,p}k\left(x,x'\right)-2E_{p,q}k(x,y)+E_{q,q}k\left(y,y'\right) \tag{5.1.2}$$

Aplicant de nou la idea anterior que dues distribucions si els seus moments són iguals, els autors empren la següent mesura per a quantificar la distància entre dues distribucions p i q on $k(\mathring{u},\mathring{u})$ es tracta de qualsevol kernel positiu:

$$\begin{aligned} D_{\mathrm{MMD}}(q\|p)=\mathbb{E}_{p(z),p(z')}\left[k\left(z,z'\right)\right]-2\mathbb{E}_{q(z),p(z')}\left[k\left(z,z'\right)\right] \\ +\mathbb{E}_{q(z),q(z')}\left[k\left(z,z'\right)\right] \end{aligned} \tag{5.1.3}$$

On $D_{\mathrm{MMD}}=0$ si i només si $p=q$

5.2 Modificació de la funció de pèrdua

Els autors d'aquest VAE proposen la modificació de la funció de pèrdua per tal de millorar la generació de noves dades i evitar els problemes mencionats anteriorment. Aquesta nova funció no utilitzaria el concepte de $ELBO(\theta,\phi)$ sinó que reemplaçaria la funció objectiu original:

$$\log P(x)=ELBO(\phi,\theta)+\mathcal{D}_{KL}\left(Q_\phi(z\mid x)||P_\theta(z\mid x)\right) \tag{5.2.1}$$

On abans el nostre $ELBO(\phi,\theta)$ era:

$$ELBO(\phi,\theta)=E_{z\sim Q_\phi}\left[-\log Q_\phi(z\mid x)+\log P_\theta(x,z)\right]= \tag{5.2.2}$$

$$-\mathcal{D}_{KL}\left(Q_\Phi\left(z\mid x^{(i)}\right)\|P_\theta(z)\right)+E_{z\sim Q_\phi}\left[\log P_\theta\left(x\mid z\right)\right] \tag{5.2.3}$$

Ara serà reemplaçat per:

$$\mathrm{MMD}-\mathrm{VAE}(\phi,\theta)=\mathrm{MMD}(Q_\phi(z)\mid P(z))+\mathbb{E}_{P_{data}(x)}\mathbb{E}_{Q_\phi(z|x)}\left[\log P_\theta(x\mid z)\right] \tag{5.2.4}$$

De forma que la nostra funció objectiu passarà a ser:

$$\log P(x)=\mathrm{MMD}-\mathrm{VAE}(\phi,\theta)+\mathcal{D}_{KL}\left(Q_\phi(z\mid x)||P_\theta(z\mid x)\right) \tag{5.2.5}$$

Part II

Implementació

Capítol VI

Creació d'un autoencoder variacional

6.1 Definició

Un cop hem establert els fonaments teòrics on es dóna suport al VAE s'introdueix ara a la part més pràctica del treball. En els següents apartats es detallarà el procés per tal de construir els nostres VAEs així com els elements i estructures utilitzades al llarg del treball.

6.2 Proposta de treball i descripció de les dades emprades

L'objectiu principal de la part experimental del projecte és aprofundir en el coneixement del autoencoder comprenent així com es poden traduir les anterior formulació matemàtiques a un model generatiu real que sigui capaç de generar noves observacions.

Es va decidir l'ús del dataset *MNIST* per tal de comprovar el funcionament i els resultats dels nostres models. Tots els models s'executen amb les mateixes dades per tal de facilitar la comparació entre models eliminant qualsevol classe de variabilitat introduïda en el moment d'utilitzar dades diferents.

El dataset *MNIST* consta d'un total de 70.000 exemples de números escrits a mà. 60.000 d'aquests exemples es troben en train set i la resta es troben al test set. Les característiques d'aquest conjunt de dades – totes les imatges tenen la mateixa mida, els números estan ben definits, centrats i escalats correctament- han fet que es converteixi el més utilitzat en el món del deep learning a l'hora d'entrenar models i comprovar que

funcionen correctament.

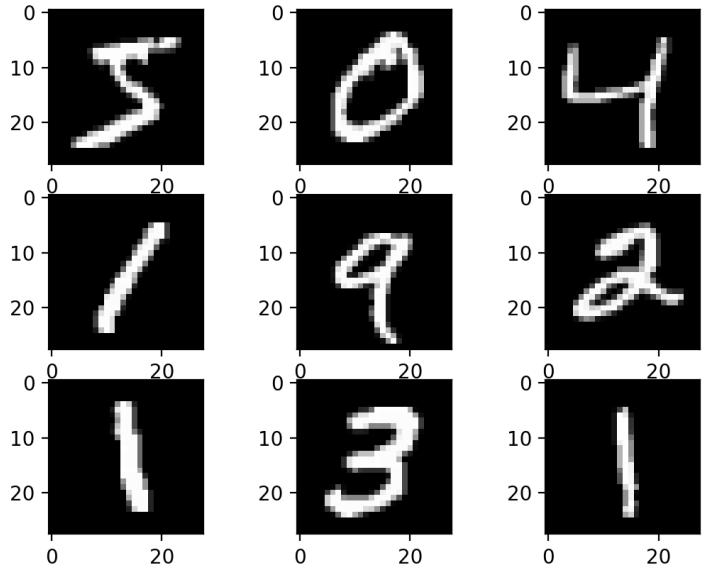


Figura 6.1: Mostra del dataset *MNIST*
(Font: Chris Burges 2010)

6.3 Arquitectura dels autoencoders variacionals

L'estructura interna de les xarxes neuronals pot ser clau a l'hora d'obtenir bons resultats i aconseguir una arquitectura eficient. Existeixen vàries propostes respecte com construir les capes que formen l'autoencoder variacional.

L'opció final va ser crear tres VAEs, dos d'ells comparteixen tots els elements que els conformen però presenten una arquitectura de capes diferent. L'altre VAE que ja hem presentat anteriorment com MMD-VAE empra una arquitectura convolucional però utilitza una funció objectiu diferent. No obstant això, els tres models comparteixen l'arquitectura elemental descrita anteriorment: es basen en un encoder, un espai latent i un decoder.

6.3.1 VAE

- **Encoder**
  El primer cas es tracta del autoencoder variacional més bàsic proposat per l'autor original. L'encoder es tracta d'un perceptró multicapa que utilitza una primera

capa amb tantes neurones com píxels té una imatge, seguit de dues capes amb 512 neurones que tracten d'examinar i aprendre les relacions intrínseques dels píxels. L'última capa del decoder consisteix en un total 2 x dimensió de l'espai latent de neurones.

S'ha decidit per tant definir el nostre encoder com:

$$Q_\phi(z|x) = N(\mu(X), \Sigma(X)) \tag{6.3.1}$$

On els paràmetres μ, Σ d'aquesta distribució son els valors de l'última capa.

- **Decoder**
  El nostre decoder està format en aquest cas per una capa d'oculta que té les mateixes dimensions que el nostre espai latent, dues capes ocultes que contenen un total de 512 neurones cadascuna i una capa de sortida amb 784 neurones, cada una associada a un píxel de la nostra imatge generada.

Recollim l'estructura del VAE implementat en la següent taula:

Tipus de capa	**Output**	**Característiques de la capa**
Input	5	batch size = 100, input size = 784
Dense	12	#hidden units = 128, activation = "linear"
Dense	73	#hidden units = 128, activation = "linear"
Dense	73	#hidden units = 128
Mostreig de z		
Input	73	input size = 784
Dense	73	#hidden units = 128, activation = "linear"
Dense	73	#hidden units = 128, activation = "linear"

Taula VI.1: Estructura del VAE standard

6.3.2 Conv-VAE

- **Encoder**
  La gran diferència en aquest apartat radica en la utilització de capes convolucionals en comptes de l'original perceptró multicapa. Aquestes capes han demostrat ser realment eficients quan es treballa amb imatges.
  S'especifica la grandària de les imatges en la capa d'entrada que permetrà el workflow

de la nostra xarxa. A continuació, és s'afegeixen dues capes convolucionals on s'aplica un filtratge i un pooling determinat a cadascuna d'aquestes per tal d'extreure l'estructura de les dades. Aquestes capes convolucionals confluiran a una capa *Dense* per tal de retornar la mitjana i variància de la nostra distribució com és habitual.

- **Decoder**
  Després de mostrejar la nostra distribució normal, aquests valors es passaran al decoder que ara consistirà en un seguit de capes convolucionals que acabaran retornat la nostra imatge generada en el format original.

Mostrem l'estructura completa detallada en la següent taula:

Tipus de capa	**Output**	**Característiques de la capa**
Input	5	batch size = 100, width = s,height = s, #channels = 1,activation = "relu"
Conv2D	12	#filters = 32, kernel size = 3, padding = "valid", strides = 2,activation = "relu"
Conv2D	73	#filters = 64, kernel size = 3, padding = "valid", strides = 2, activation = "relu"
Flatten	73	Reorganitza les neurones en format allargat
Dense	73	#hidden units = 128
Split	73	#num splits = 2
Mostreig de z		
Dense	73	#hidden units = 1568, activation = "relu"
Reshape	(7, 7, 32)	Reorganitza les neurones en format convolucional
Conv2D	73	#filters = 64, kernel size = 3, padding = "same", strides = 2, activation = "relu"
Conv2D	73	#filters = 32, kernel size = 3, padding = "same", strides = 1
Conv2D	73	#filters = 1, kernel size = 3, padding = "same", strides = 1

Taula VI.2: Estructura del Conv-VAE

L'estructura de les capes és l'única diferencia respecte al nostre VAE estàndard. La funció objectiu, les funcions d'activacions i la resta d'elements es mantindran igual per aquest autoencoder variacional.

6.3.3 MMD-VAE

- **Encoder**
  De forma similar al Conv-VAE, definim el nostre encoder utilitzant les capes convolucionals. La gran diferència ara radica en el fet que el nostre encoder no retorna dos paràmetres a diferència dels altres dos encoders.

- **Decoder**
  El decoder que ara consisteix en diverses capes convolucionals que retornaran la imatge al format original de forma similar als dos anteriors.

Tipus de capa	**Output**	**Característiques de la capa**
Input	5	batch size = 100, width = s,height = s, #channels = 1,activation = "relu"
Conv2D	12	#filters = 32, kernel size = 3, padding = “valid”, strides = 2,activation = "relu"
Conv2D	73	#filters = 64, kernel size = 3, padding = “valid”, strides = 2, activation = "relu"
Flatten	73	Reorganitza les neurones en format allargat
Dense	73	#hidden units = 128
Mostreig de z		
Dense	73	#hidden units = 1568, activation = "relu"
Reshape	(7, 7, 32)	Reorganitza les neurones en format convolucional
Conv2D	73	#filters = 64, kernel size = 3, padding = “same”, strides = 2, activation = "relu"
Conv2D	73	#filters = 32, kernel size = 3, padding = “same”, strides = 1
Conv2D	73	#filters = 1, kernel size = 3, padding = “same”, strides = 1,activation = "sigmoid"

Taula VI.3: Estructura del MMD-VAE

6.3.4 Funció objectiu

La funció que buscarem optimitzar tal com s'ha descrit en l'apartat de fonaments teòrics tant pel VAE estàndard així com pel Conv-VAE és:

$$\log P(X) - \mathcal{D}[Q(z \mid X) \| P(z \mid X)] = E_{z \sim Q}[\log P(X \mid z)] - \mathcal{D}[Q(z \mid X) \| P(z)] \quad (6.3.2)$$

Pel que fa el MMD-VAE tenim:

$$\log P(x) = \text{MMD} - \text{VAE}(\phi, \theta) + \mathcal{D}_{KL}\left(Q_\phi(z \mid x) || P_\theta(z \mid x)\right) \quad (6.3.3)$$

On $MMD - VAE(\phi, \theta)$ és pot expressar com:

$$\text{MMD} - \text{VAE}(\phi, \theta) = \text{MMD}(Q_\phi(z) \mid P(z)) + \mathbb{E}_{P_{data}(x)} \mathbb{E}_{Q_\phi(z|x)} \left[\log P_\theta(x \mid z)\right] \quad (6.3.4)$$

6.4 Components del model

A continuació es detallen un llistat d'elements que comparteixen els tres models. La implementació tant com el concepte darrere dels següents elements és idèntic pels tres models.

6.4.1 Espai lantent

Aquest punt es tracta des d'una perspectiva experimental. És difícil determinar a priori la dimensió de l'espai latent a utilitzar. És clar que únicament dues dimensions poden semblar insuficients per tal d'obtenir una representació latent informativa de les nostres dades, però tampoc sembla clar fins a quin punt el nostre model pot extreure informació d'un espai latent massa complex definit en un espai d'una alta dimensió.

És per això que es decideix el tractament d'on projectarem les nostres dades com un hiperparàmetre. Provarem diferents valors utilitzant el conjunt de validació de dades amb l'objectiu d'observar amb quin valor s'obté el valor mínim per la nostra funció objectiu.

6.4.2 Optimitzadors

D'una forma similar a l'estratègia seguida per determinar la dimensió final de l'espai latent i donat la gran varietat d'algoritmes que es poden utilitzar, es decideix tractar l'optimitzador com un hiperparàmetre el qual s'escollirà un cop s'hagi realitzat una cerca intensiva per tal d'observar quin és el valor que minimitza la funció de pèrdua. Així mateix, el learning rate serà tractat de la mateixa forma i s'escollirà a través dels resultats empírics.

6.4.3 Funcions d'activacions utilitzades

S'utilitzen les mateixes funcions d'activacions que Kigma va defensar al seu treball per tal de mantenir continuar amb la idea original que va aconseguir provar uns resultats excel · lents en la generació d'exemples. En totes les capes del autoencoder s'empra funció d'activació ReLu per tal d'optimitzar la convergència, en l'última capa però donat que volem obtenir els valors per la nostra distribució normal, s'aplica la funció d'activació lineal.

Respecte al decoder, s'utilitza durant tota l'estructura la funció d'activació *ReLu*. L'última capa del decoder no utilitza la funció sigmoide com es podria esperar donat que volem retornar valors entre 0 i 1 però aplicarem la funció fora un cop tinguem les prediccions per tal d'optimitzar al màxim la convergència de la nostra xarxa.

6.5 Entrenament dels models

6.5.1 Procés d'entrenament

El procés d'entrament dels models ha sigut el següent. S'han completat un total de 50 iteracions (epochs) dins de cada iteració, el dataset es dividia en conjunts de 100 imatges aleatòries (mini-batch) per cada mini-batch completat, s'actualitzaven els paràmetres de la nostra xarxa amb l'ús de l'optimitzador corresponent..

6.5.2 Hiperparàmetrització

Tal com s'introduïa abans, per tal dels nostres models siguin capaços de generar els resultats òptims, s'ha creat un hypergrid de cerca intensiva. La intuïció darrera d'aquest recurs es basa en el concepte conegut com hypertuning; un cop que s'han provat totes les combinacions que volem estudiar per uns determinats paràmetres, s'escull la configuració òptima que minimitza la funció de objectiu dels models en una nova divisió del dataset que s'anomena conjunt de validació.

Dimensió de l'espai latent	**Optimitzador**	**Tasa d'aprenentatge**
2,20,50	adadelta, adam, rmsprop	1e-04,7e-04

Taula VI.4: Hiperparàmetres estudiats durant el projecte

Remarcar que aquest procés és especialment costós computacionalment parlant donat

que és s'entrenen un total de $3 \times 3 \times 2 = 18$ models i aquest procediment es replica pels tres models. En mitjana, el temps en completar totes les iteracions fins a entrenar completament un model és proper a 1 h.

6.6 Eines emprades en la implementació

6.6.1 Hardware

Respecte als recursos computacionals, Tensorflow ens dóna la possibilitat de treballar amb la nostra GPU en comptes d'utilitzar el tradicional CPU. Aquesta unitat de processament aconsegueix reduir el temps de càlcul considerablement tot i que la seva instal · lació afegeix una dificultat més. L'ús de la GPU només ha sigut necessari en el moment de tunejar els paràmetres dels models.
Els models s'han entrenat en un ordinador amb les següents característiques:

- Processador: Intel(R)Core(TM) i7-6700HQ CPU @2.60GHz 2.59 GHz
- Targeta gràfica (GPU): Nvidia GTX 1060 3GB
- Memòria RAM: 16GB (15.9GB utilitzables)

6.6.2 Software

Els models s'han construït utilitzant una de les propostes més clàssiques i que més potencial ha demostrat al llarg dels últims anys. Es treballa utilitzant el framework de Tensoflow desenvolupat per Google que va ser presentat el 2015 com una llibreria de software obert que es basava en uns objectes anomenats tensors. Aquests objectes poden ser pensats com matrius N-dimensionals i tenen com a objectiu facilitar l'entrenament dels models neuronals.

Existeixen dues formes principals de programar amb Tensorflow, la versió eager que realitza els càlculs de forma seqüencial, on es pot imprimir per pantalla els valors dels tensors computats, o el mode estàndard on els tenors passen a ser elements més abstractes i no és tan senzill accedir a aquests. El projecte està creat utilitzant la versió eager, que és la versió per defecte de TF 2.0.

En combinació amb Tensorflow, s'afegeix la llibreria keras, un altre projecte bastant nou que facilita la creació dels models a través d'un codi net i senzill per tal de crear les diferents capes que inclou el nostre model. El codi emprat per generar les noves observacions així com els grids creats amb el suport de TensorBoard, una extensió de

Tensorflow que ens permet visualitzar les diferents execucions, és pot trobar en següent repositori de *GitHub*:

https://github.com/adrianvizoso/VAE_TFG

6.7 Resultats experimentals

En vista dels resultats observats en les tres execucions 7.11 es conclou que la millor configuració pels models VAE i Conv-VAE són:

Models	Paràmetres		
	Dimensió latent	Optimitzador	Tassa d'aprentatge
VAE	20	Adam	0.0001
Conv-VAE	20	Adam	0.0001
MMD-VAE	20	Adam	0.00075

Taula VI.5: Resultats obtinguts després del *grid-search*

Utilitzarem ara l'API TensorBoard (7.3) pel següent anàlisi. Aquests resultats indiquen un dels grans punts a tenir en compte a l'hora de crear els nostres VAEs. Dos dimensions semblen ser insuficients per crear un espai latent ric en informació i 50 no milloren el resultat. 20 relacions semblen ser el punt intermedi que permeten recollir les característiques i relacions principals deel nostre conjunt de dades, millorant així la nostra funció objectiu

És especialment interessant notar com els resultats són idèntics pels VAEs que tenen la mateixa funció objectiu. Això és un clar indicador de com aquesta funció és un clar condicionant a l'hora d'entrenar els nostres models. Tot i que, en el cas del MMD-VAE l'estructura és similar, aquest necessita una tassa d'aprenentatge superior per tal d'assolir el mínim en la funció objectiu. Per últim, és important mencionar que Adam és un dels algoritmes més eficients i així ho demostra en aquesta hiperparàmetrització, sent escollit pels tres models com millor optimitzador.

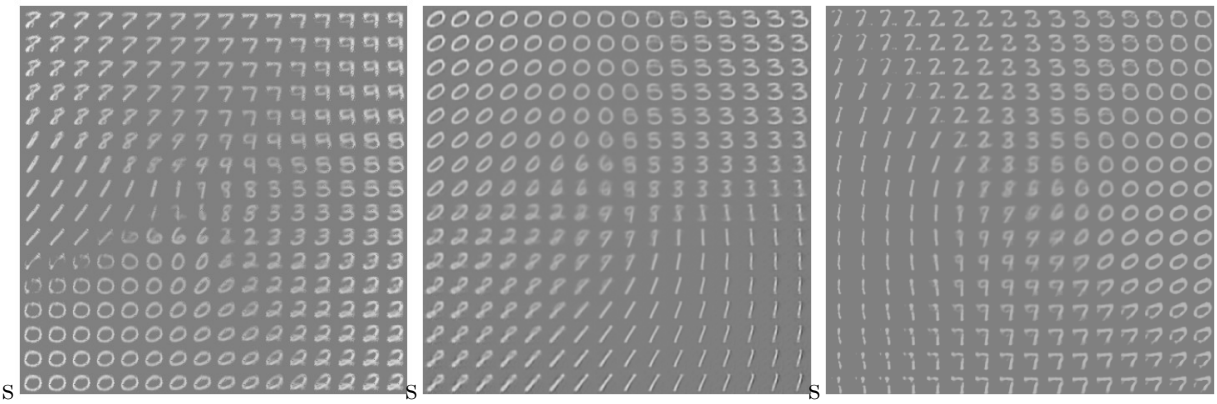
6.8 Anàlisis dels models

A continuació es proposa l'estudi dels models creats amb els paràmetres escollits anteriorment. Per tal d'avaluar els models es creen tres elements claus que ens permetran obtenir una idea de que efectiu és el nostre model i detectar possibles problemes. Aquests elements són la representació de l'espai latent, el grid i els *outputs* generats pel nostre model.

6.8.1 Anàlisis del grid

El grid és un element generat després d'entrenar el nostre model per tal d'observar l'acompliment en les diferents iteracions. Obtenim els percentils d'una distribució gaussiana multivariant de paràmetres $N(0, 1)$ través del mostreig de la funció de distribució inversa de probabilitat (CDF), un cop tenim aquests valors, els utilitzem com valors d'entrada pel nostre decoder, que s'encarregarà de generar una imatge en el pla 2-dimensional.

Les següents imatges mostren les imatges generades pel nostre decoder entrenat utilitzant el dataset *MNIST*. S'observen com alguns dígits central és mostren borrosos i no són prou distingibles. Molts d'aquests símbols no definits són combinacions de dos dígits que donen lloc a una representació confusa. Aquest comportament està clarament lligat a l'espai latent del nostre model i la representació del dígits que s'extreu més en aquest punt en l'apartat següent.



(a) VAE (b) Conv-VAE (c) MMD-VAE

Figura 6.2: Grid dels diferents models
[Font: Elaboració propia]

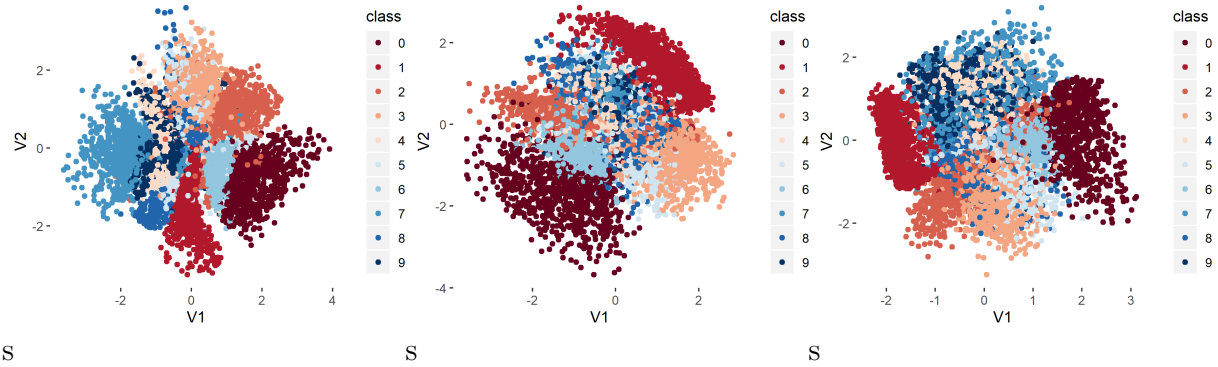
6.8.2 Distribució de l'espai latent

Observem l'espai latent generat pel nostre model a través de la representació en $\mathbb{R}^2$ de les nostres variables latents.

És important recordar que per tal que el nostre decoder pugui generar unes mostres similars als dígits del nostre conjunt de dades sense cap anomalia persistent , busquem un espai latent homogeni sense discontinuïtats i distribuït uniformement, on cada dígit hauria d'estar representat a una regió tancada i definida, cadascuna hauria d'ocupar el

mateix espai i tenir una forma similar.

S'ha identificat cada punt amb el dígit corresponent per tal d'entendre el comportament del model: volem observar quins dígits són més problemàtics i quins, al contrari, es projecten en aquest espai com un conjunt tancat i ben definit. La Figura 5.4 mostra aquestes visualitzacions dels nostres dos conjunts de proves de dades latents per *MNIST*. L'estructura de l'espai latent és diferent pel VAE estàndard i pel Conv-VAE, però tots dos mostren un comportament similar.



(a) VAE (b) Conv-VAE (c) MMD-VAE

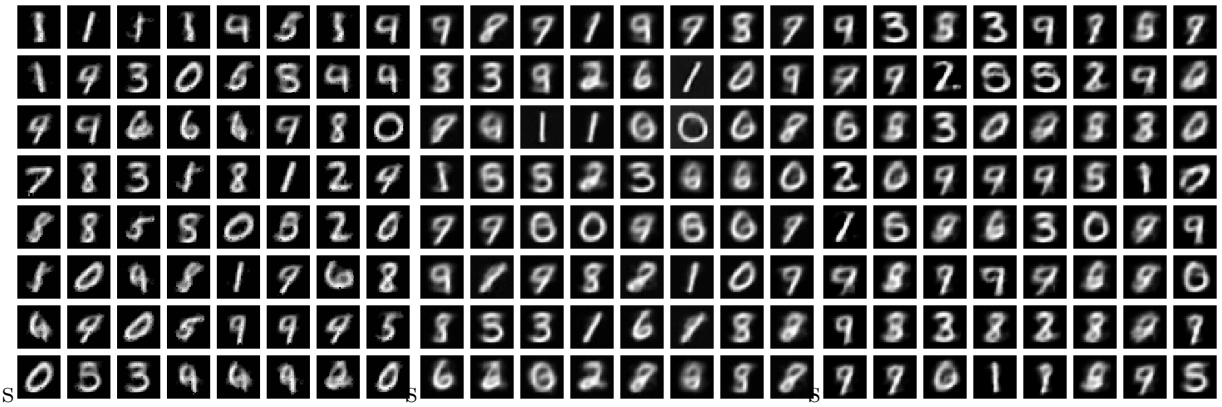
Figura 6.3: Distribució del espai latent per cada model
[Font: Elaboració propia]

Podem veure que els tres VAEs, inclosos aquells que comparteixen la funció objectiu, estructuren les classes de forma diferent però tots tres presenten algunes similituds. Les representacions en l'espai latent semblen estar ben definides, en grups delimitats i àrees contínues. No obstant això, encara hi ha regions amb gran superposició entre classes, i algunes classes es concentren en petites regions de l'espai latent en comparació amb d'altres. És rellevant el fet que els tres models tenen dificultats per separar els dígits vuit i nou. Es pot observar com aquests nombres no tenen una àrea tan ben definida com el zero o l'u. Aquest patró, que es repeteix als tres models, dóna lloc al grid observat anteriorment, on la transició entre els números que tenen una representació més pobre és més brusca i alguns cops dóna lloc a xifres difícils de reconèixer.

Tot i això, els tres VAE sembla que, certament, han après una codificació de les dades útil per la generació de més mostres. Encara així, es remarca que aquests models han estat entrenats amb únicament dues dimensions i és possible que una representació més complexa en un espai dimensional més alt pugui afavorir al model obtenint un espai latent més informatiu i una codificació capaç de recollir les relacions més profundes entre els píxels.

6.8.3 Imatges generades

Per últim observem a continuació les imatges generades pels diferents models. Aquestes han estat generades amb la combinació optima de hiperpàrametres sense limitar l'espai latent a dues dimensions. Es generen 64 imatges per cada model per tal de tenir una visió global de l'efectivitat dels models a l'hora de generar noves imatges.



(a) VAE (b) Conv-VAE (c) MMD-VAE

Figura 6.4: Imatges generades pels diferents models.
[Font: Elaboració propia]

Es pot observar com la qualitat de les imatges sembla ser molt similar i és difícil identificar grans diferències. El Conv-VAE sembla donar les imatges més borroses i un dels models més propensos a generar dígits no delimitats com combinacions de dos nombres com és el zero que sembla contenir un el número u dins. El VAE estàndard sembla millorar la qualitat d'aquests nous nombres generats però també podem observar alguns dígits que no queden definits. El MMD-VAE no sembla solucionar tampoc aquesta problemàtica i també genera alguns casos de difícil identificació i amb una qualitat similar al Conv-VAE.

A pesar dels punts anteriors, hem aconseguit el nostre objectiu: hem generat mostres noves a partir d'un conjunt de dades. Encara que alguns exemples presentats anteriorment siguin problemàtics, podem veure com efectivament hem sigut capaços de generar nous dígits com a combinacions a partir d'altres.

6.9 App de visualització de resultats

Utilitzant el paquet de *Shiny* s'ha desenvolupat una *app* interactiva que permet observar com el nostre model aprèn a mesura que van augmentar les *epoch*. Aquesta aplicació permet escollir entre els tres models estudiants durant el projecte i conté tres grans funcionalitats per a cada model:

6.9.1 Imatges generades durant el procès d'entrenament.

En aquesta primera secció podem observar les imatges que el model va generar durant el procés d'entrenament.

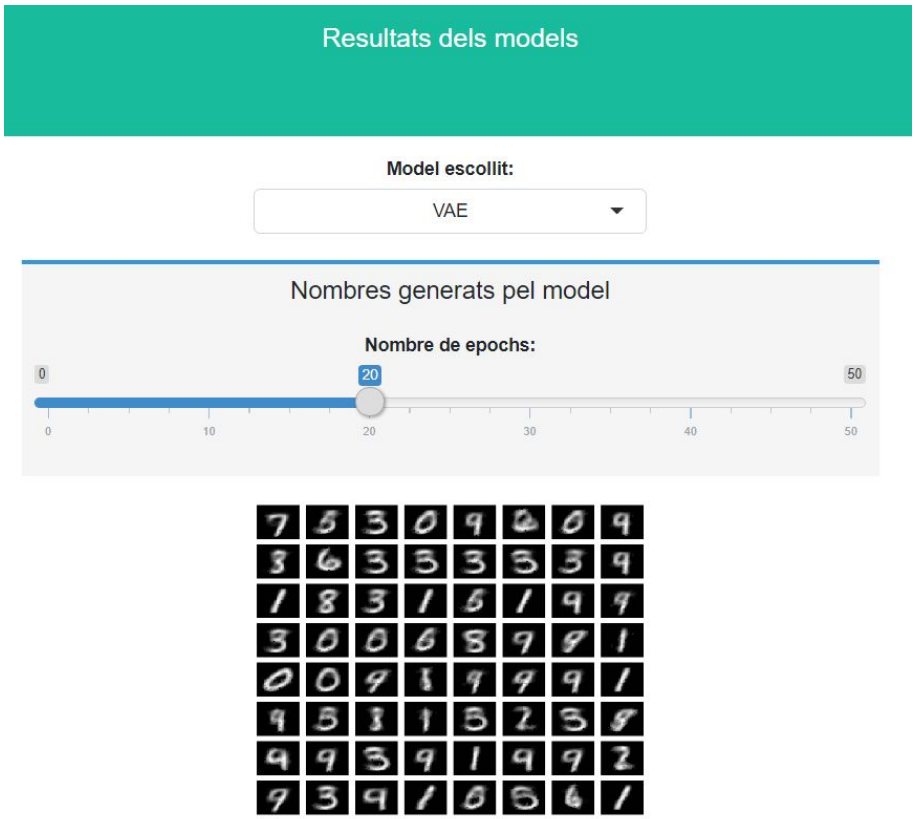


Figura 6.5: Procès de generació de noves mostres.
[Font: Elaboració propia]

Durant les diferents *epochs* el model va generant dígits cada cop més clars i reconeixibles. A mesura que el model va aprenent l'estructura de les dades, els nombres tendeix a estar més delimitats i no presenten tantes irregularitats.

6.9.2 Visualització de l'espai latent

Aquesta secció mostra com l'espai latent va evolucionant a mesura que les iteracions van avançant. A través d'aquesta funcionalitat podem entendre com el nostre model va consolidant l'espai latent i quina forma va adquirint durant el procés. Observem que durant les primeres iteracions l'espai latent presenta una forta variabilitat, que es va reduint a mesura que fins que en les últimes iteracions no es pot observar gairebé cap canvi apreciable.

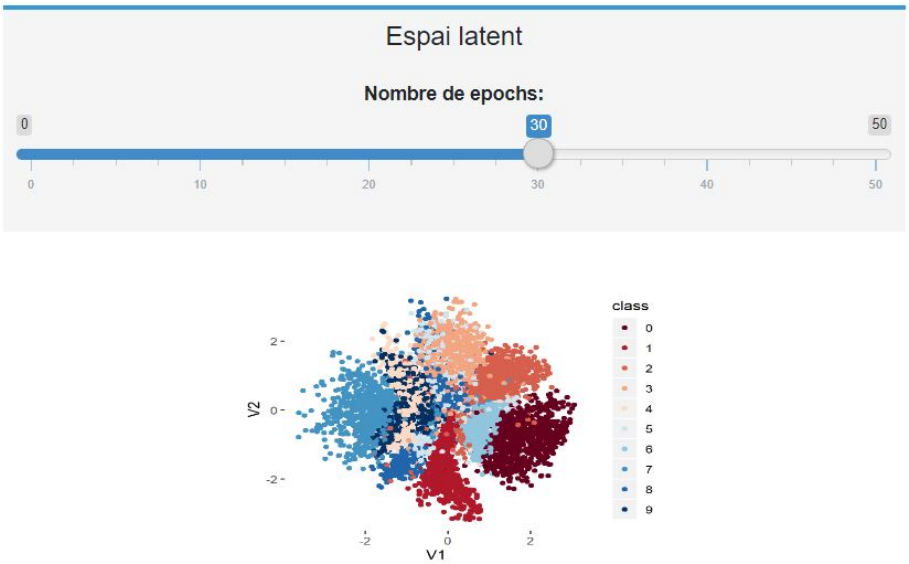


Figura 6.6: Procès de generació de noves mostres.
[Font: Elaboració propia]

6.9.3 Evolució del grid

De forma similar a l'espai latent, podrem utilitzar aquesta funcionalitat per comprendre l'evolució del grid. Fent ús del grid observarem quins elements pot generar el VAE més detalladament i com es produeix la transacció d'un dígit a un altre. Aquest element però, s'explica en més detall en la secció 6.8.1 .

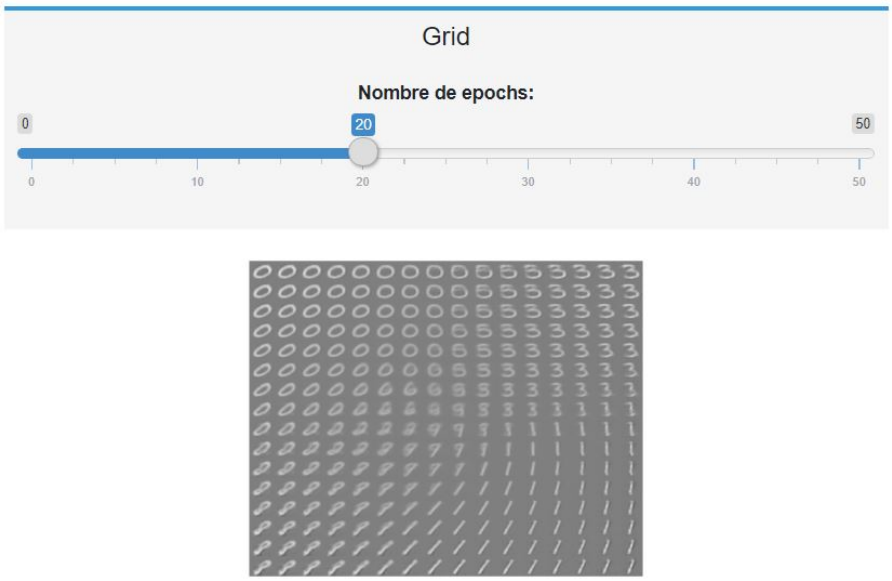


Figura 6.7: Procès de generació de noves mostres.
[Font: Elaboració propia]

6.9.4 Generació de nou dígits

Un cop el nostre model ha estat entrenat utilitzant els millors hiperparàmetres i ha completat totes les *epoch* el VAE està preparat per generar nous dígits. Tensorflow ens permet guardar un *checkpoint* del nostre model en aquest punt, que ens serà de gran utilitat per carregar-ho més tard i poder generar noves imatges en viu prement el botó *"Generar nou dígit"*.

Figura 6.8: Procès de generació de noves mostres.
[Font: Elaboració propia]

Part III

Conclusions

Capítol VII

Conclusions i propostes de continuació

7.1 Conclusions

Aquest treball neix amb l'objectiu de comprendre que és un autoencoder variacional, reflexionar sobre la seva estructura i conèixer quines variacions existeixen sobre aquest. La primera variació que dóna lloc al Conv-VAE es basa en la modificació de l'arquitectura de la xarxa neuronal que modelitza els dos principals elements dels VAE – l'encoder i el decoder – a través de la introducció de les capes convolucionals.

Seguidament, es reflexiona sobre els principals problemes intrínsecs del VAE. Es prova que els fonaments en els quals es basa el VAE permeten optimitzar la seva funció objectiu fins i tot quan els paràmetres dels models no són estables, i es mostra com en alguns casos l'espai latent no és tan informatiu com es podria pensar.
Per tal de resoldre els problemes mencionats, es modifica la funció objectiu del VAE, introduint el kernel d'una distribució com a mesura de discrepància. Aquest enfocament passa a ser la segona modificació que s'estudia en el projecte i l'anomenarem MMD-VAE.

La següent part del treball és eminent pràctica. S'utilitza un dels *framework* més famosos per tal d'implementar el VAE original i les variacions mencionades anteriorment. Per tal de trobar la configuració òptima es realitzà una cerca intensiva dels hiperparàmetres claus del model. Aquest enfocament no ha permès únicament trobar aquella configuració que genera les millors imatges sinó que també aconsegueix aprofundir en la comprensió del funcionament aquests models i quina influència tenen aquests paràmetres a l'hora de minimitzar la funció de pèrdua. Per tal de visualitzar aquests resultats d'una forma

interactiva, es crea una *app* que permet veure la convergència dels models i com són capaços de generar noves mostres a partir d'un conjunt de dades.

Es conclou per tant que s'hi han aconseguit els objectius de la proposta d'investigació que es va fer al principi. S'ha reflexionat sobre el VAE des de la perspectiva més clàssica com a xarxa neuronal així mateix com a model probabilístic. S'han estudiat en detall els problemes que aquest model planteja, s'ha proposat una solució alterativa a aquests i finalment s'ha realitzat una implementació que no ha presentat grans problemes.

7.2 Propostes de continuació

Com a propostes de continuació, seria molt interessant aprofundir en enfocaments novells com són el β-VAE. També són variacions d'alt interès totes aquelles que treballen noves propostes sobre la distribució a priori del VAE, a pesar de les limitacions que aquestes presenten.

També pot ser convenient l'estudi d'aquests models des d'una perspectiva més pràctica. Una de les grans aplicacions dels VAEs és la detecció d'anomalies, un cap d'estudi molt actiu i interessant en les ciències de la computació actualment.

Part IV

Annexos

7.3 *TensorBoard*

7.3.1 Què és *TensorBoard* i què ens ofereix?

Tensorboard és una eina de visualització de resultats que forma part de Tensorflow. Aquesta interfície és especialment útil quan es tracta de comprendre el funcionament, observar si existeix algun problema i optimitzar els models creats.

Ens centrarem en tres *tabs* principals a l'hora de seleccionar els millors hiperparàmetres i comprendre com aquests afecten la funció objectiu.

7.3.1.1 *Scalars*

En aquest apartat podem observar tota la informació necessària per escollir els nostres hiperparàmetres. És aquí on es recullen les nostres mètriques sobre la funció de pèrdua i on podrem observar detalladament com fluctua. Ens interessa especialment assegurar-nos que la nostra funció de pèrdua convergeix i un cop el nostre model ha après la informació necessària, aquesta es manté estable.

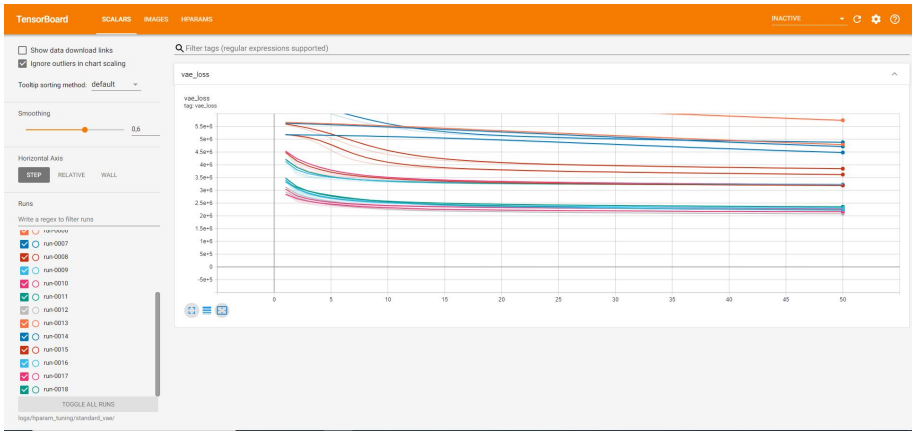


Figura 7.1: Captura de pantalla de l'aplicatiu TensorBoard de l'apartat *scalars* [Font: Elaboració propia]

7.3.1.2 *Hparams*

Aquesta secció permet comprendre quin és l'efecte marginal del cadascun dels hiperparàmetres i com afecten aquests a la nostra funció objectiu.

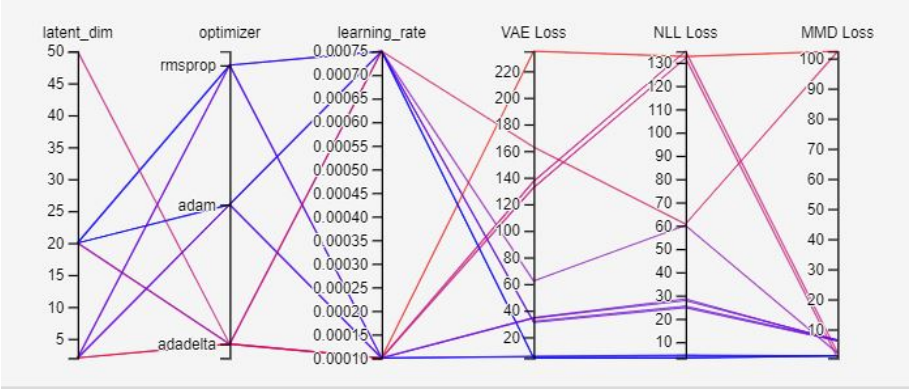


Figura 7.2: Captura de pantalla de l'aplicatiu TensorBoard de l'apartat *hparams*. En aquesta figura és pot observar la infuència de cada hiperparàmetre de forma marginal sobrel el valor total de la funció de pèrdua.

[Font: Elaboració propia]

7.4 Inequació de Gibbs

Definició 3 (Inecuació de Gibbs) *Suposem que, $P = \{p_1, \ldots, p_n\}$ és una distribució de probabilitat qualsevol. Llavors per qualsevol altra distribució de probabilitat $Q = \{q_1, \ldots, q_n\}$ es compleix sempre la següent inequació: @bremaud2012introduction*

$$-\sum_{i=1}^{n} p_i \log p_i \leq -\sum_{i=1}^{n} p_i \log q_i \tag{7.4.1}$$

on la igualtat és dona si i només si $p_i = q_i$ per tot i.

Amb altres paraules, l'entropia d'una distribució P és menor o igual que l'entropia amb qualsevol altra distribució Q.

7.5 Desigualtat triangular

El teorema de desigualtat triangular afirma que en qualsevol triangle la longitud d'un dels costats no pot mai superar a la suma de les longituds dels altres dos.

El teorema pot generalitzar-se a espais vectorials normats, obtenint-se la següent versió de la desigualtat triangular:Hardy, Littlewood, and Pólya (1967)

Definició 4 (Desigualtat triangular) *En tot espai vectorial normat V, podem dir:*

$$V \forall x, y \in V, \quad \|x + y\| \leq \|x\| + \|y\| \tag{7.5.1}$$

7.6 Problemes de classificació binària

En un problema de classificació binària l'objectiu és ser capaços d'assignar una nova observació a una categoria. Un exemple seria assignar a una imatge la classificació de gos o gat depenent l'animal que contingui. Per realitzar aquesta classificació s'entrena el model a partir d'un conjunt de dades que contenen la informació sobre a quina classe pertanyen i després de construir una funció separadora entre classes es classifica aquesta nova observació en funció del valor obtingut per aquesta funció. Existeixen dos tipus de problemes de classificació, els problemes linealment separables on totes les observacions es poden separar utilitzant una recta o un hiperplà i els problemes on calen funcions més complexes. Kröse et al. (1993)

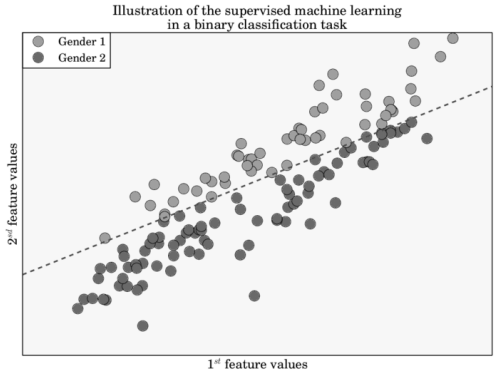


Figura 7.3: Exemple de classificació binària. La figura mostra com separar dos classes utilizant un hiperplà.

(Font: Pan et al. 2019)

7.7 Aprenentatge supervisat

En un problema d'aprenentatge supervisat, disposem d'un parell d'elements (x, y) on x és un vector que pertany a l'espai $\mathbb{R}^m$ que conté les dades d'entrada del model i y és el valor numèric o l'etiqueta de classe que volem predir.

L'objectiu en aquest tipus de problema és ser capaços d'aprendre o aproximar una funció $f(X)$ que ens permeti establir una relació entre les variables d'entrades i el resultat desitjat.

Alguns exemples d'aquest tipus de problema són: els problemes de classificació, els problemes de regressió o la detecció d'objectes, entre d'altres.

7.8 Aprenentatge no supervisat

En aquest tipus de problema, a diferència dels problemes d'aprenentatge supervisat, no disposem del valor real de les dades d'entrada al model.

El nostre objectiu en aquest tipus de problemes és ser capaços d'aprendre l'estructura oculta de les dades. El *clustering*, la reducció la dimensió o l'estimació de la densitat de les dades són alguns exemples d'aquest tipus de problemes.

7.9 Xarxa generativa antagònica (GANs)

És interessant observar un altre enfoc al problema que tractarem sobre la generació de noves dades. Els models GANs és l'alternativa clàssica als autoencoders variacionals. Es tracta d'un dels principals models generatius de dades que s'utilitzen actualment, basat principalment en l'entrenament de dos models d'aprenentatge profund per separat.@mirza2014conditional

El primer model s'anomena generador i s'encarrega de generar mostres de la distribució plausibles. L'altra xarxa neuronal s'anomena discriminador i tracta de diferenciar les mostres generades dels exemples reals.

Aquests dos models s'enfronten des d'un punt de vista de la teoria de jocs, on la generativa crea noves imatges i les barreja amb imatges reals. Per altra banda, la xarxa discriminadora tracta de decidir si cada imatge pertany o no al conjunt de dades d'entrenament. Cada cop que el discriminador detecta una diferència s'ajusten els paràmetres de les dues xarxes per separat.

En aquest procés el generador és capaç d'aprendre a generar mostres realment realistes de la distribució de les dades. Això ens permet utilitzar aquesta xarxa per a generar noves dades que podrien semblar totalment reals als ulls de qualsevol persona. Salimans et al. (2016)

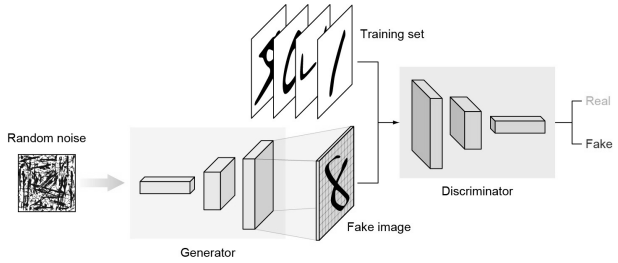


Figura 7.4: Procès esquemàtic del model generatiu GAN.
[Font: Pan et al. (2019)

7.10 Demostració Discrepancia Màxima Mitjana

Definició 5 *$D(p, q, \mathcal{F}) = 0$ iff $p = q$, when $\mathcal{F} = \{f \mid \|f\|_{\mathcal{K}} \leq 1\}$ és una bola unitaria en un espai de Hilbert de Kernel Reproduïble.*

Tal com es prova a Borgwardt et al. (2006) resoldre problema d'optimització següent:

$$\sup_{\|f|\leq 1} \mathbf{E}_p[f(x)] - \mathbf{E}_q[f(y)] = \sup_{\|f|\leq 1} \langle \mu_p - \mu_q, f \rangle = \|\mu_p - \mu_q\|_{\mathcal{H}} \tag{7.10.1}$$

Ens porta a la següent solució si considerem la definició inicial:

$$\begin{aligned} \|\mu_p - \mu_q\|_{\mathcal{H}}^2 =& \langle \mu_p - \mu_q, \mu_p - \mu_q \rangle \\ =& \mathbf{E}_{p,p} \langle k(x, \cdot), k(x', \cdot) \rangle - 2\mathbf{E}_{p,q} \langle k(x, \cdot), k(y, \cdot) \rangle \\ &+ \mathbf{E}_{q,q} \langle k(y, \cdot), k(y', \cdot) \rangle \\ =& \mathbf{E}_{p,p} k(x, x') - 2\mathbf{E}_{p,q} k(x, y) + \mathbf{E}_{q,q} k(y, y') \end{aligned} \tag{7.10.2}$$

D'on s'extreu l'expressió que s'utiliza en aquest treball.

7.11 Taules de resultats

Neurones en l'espai latent	Optimitzador	Tasa d'aprenentatge	Pérdua mínima
2	rmsprop	0.00075	2.1808e+6
2	adadelta	0.0001	4.4594e+6
2	adam	0.00075	2.1641e+6
2	rmsprop	0.0001	2.2206e+6
2	adadelta	0.00075	3.6106e+6
2	adam	0.0001	2.2279e+6
20	adam	0.0001	2.1061e+6
20	adadelta	0.0001	4.7290e+6
20	rmsprop	0.0001	2.3536e+6
20	adadelta	0.00075	3.8400e+6
20	rmsprop	0.00075	2.2716e+6
20	adam	0.00075	2.2597e+6
50	adadelta	0.00075	4.8727e+6
50	rmsprop	0.0001	3.2251e+6
50	adam	0.0001	3.1864e+6
50	rmsprop	0.00075	3.2035e+6
50	adadelta	0.0001	5.7189e+6
50	adam	0.00075	3.2229e+6

Taula VII.1: Taula de resultats hiperparàmetres VAE estàndar

Neurones en l'espai latent	Optimitzador	Tasa d'aprenentatge	Pérdua mínima
2	rmsprop	0.00075	2.1288e+6
2	adadelta	0.0001	3.9234e+6
2	adam	0.00075	2.1641e+6
2	rmsprop	0.0001	2.1269e+6
2	adadelta	0.00075	3.4199e+6
2	adam	0.0001	2.1129e+6
20	adam	0.0001	2.0031e+6
20	adadelta	0.0001	3.8890e+6
20	rmsprop	0.0001	2.2263e+6
20	adadelta	0.00075	3.8400e+6
20	rmsprop	0.00075	2.9985e+6
20	adam	0.00075	2.1109e+6
50	adadelta	0.00075	2.1137e+6
50	rmsprop	0.0001	3.9442e+6
50	adam	0.0001	3.0112e+6
50	rmsprop	0.00075	2.1995e+6
50	adadelta	0.0001	5.3663e+6
50	adam	0.00075	2.9319e+6

Taula VII.2: Taula de resultats hiperparàmetres Conv-VAE

Neurones en l'espai latent	**Optimitzador**	**Tasa d'aprenentatge**	**Pérdua mínima**
2	rmsprop	0.00075	30.988
2	adadelta	0.0001	234.93
2	adam	0.00075	31.848
2	rmsprop	0.0001	34.217
2	adadelta	0.00075	162.90
2	adam	0.0001	34.604
20	adam	0.0001	5.2179
20	adadelta	0.0001	137.07
20	rmsprop	0.0001	5.5812
20	adadelta	0.00075	62.210
20	rmsprop	0.00075	4.4311
20	adam	0.00075	4.3326
50	adadelta	0.00075	60.11
50	rmsprop	0.0001	5.2788
50	adam	0.0001	5.23219
50	rmsprop	0.00075	4.6755
50	adadelta	0.0001	132.39
50	adam	0.00075	4.7722

Taula VII.3: Taula de resultats hiperparàmetres MMD-VAE

Bibliografia

Ash, Robert B. 2010. *Google Books*. https://books.google.es/books?id=ngZhvUfF0UIC&pg=PA16&dq=intitle:information+intitle:theory+inauthor:ash+conditional+uncertainty&redir_esc=y#v=onepage&q=intitle%3Ainformation%20intitle%3Atheory%20inauthor%3Aash%20conditional%20uncertainty&f=false.

Bhattarai, Saugat. 2018. "Saugat Bhattarai." *A Tech Blog*. https://saugatbhattarai.com.np/what-is-gradient-descent-in-machine-learning/.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. springer.

Borgwardt, Karsten M, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. 2006. "Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy." *Bioinformatics* 22 (14): e49–e57.

Chaudhary, Mukesh. 2020. "Digit Recognizer - Mukesh Chaudhary - Medium." *Medium*. Medium. https://medium.com/@cmukesh8688/digit-recognizer-471322d4a0eb.

Chen, Xi, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. "Variational Lossy Autoencoder." http://arxiv.org/abs/1611.02731.

Chris Burges, Yann LeCun, Corinna Cortes. 2010. "MNIST Dataset." *Lecun.com*. http://yann.lecun.com/exdb/mnist/.

Doersch, Carl. 2016. "Tutorial on Variational Autoencoders." http://arxiv.org/abs/1606.05908.

Elsevier. 2020. "Fig. 2. The Comparison Between Recurrent Neural Network (Rnn) And..." *ResearchGate*. ResearchGate. https://www.researchgate.net/figure/The-comparison-between-Recurrent-Neural-Network-RNN-and-Feed-Forward-Neural-Network_fig1_338672883.

Gad, Ahmed Fawzy. 2020. "Image Compression Using Autoencoders in Keras |

Paperspace Blog." *Paperspace Blog*. Paperspace Blog. https://blog.paperspace.com/autoencoder-image-compression-keras/.

Goodfellow, Ian. 2016. "NIPS 2016 Tutorial: Generative Adversarial Networks." *arXiv.org*. https://arxiv.org/abs/1701.00160.

Hardy, GH, JE Littlewood, and G Pólya. 1967. "Inequalities. Cambridge Mathematical Library Series." Cambridge University Press.

Hoffman, Matthew D, David M Blei, Chong Wang, and John Paisley. 2013. "Stochastic Variational Inference." *The Journal of Machine Learning Research* 14 (1): 1303–47.

Huszár, Ferenc. 2015. "How (Not) to Train Your Generative Model: Scheduled Sampling, Likelihood, Adversary?" http://arxiv.org/abs/1511.05101.

Kamruzzaman, Abu, and C. C. Tappert. 2019. "Developing Deep Learning Models to Simulate Human Declarative Episodic Memory Storage." *ResearchGate*. SAI Organization. https://www.researchgate.net/publication/332540618_Developing_Deep_Learning_Models_to_Simulate_Human_Declarative_Episodic_Memory_Storage.

Kingma, Diederik P, and Max Welling. 2013. "Auto-Encoding Variational Bayes." http://arxiv.org/abs/1312.6114.

Kingma, Diederik P., and Max Welling. 2019. "An Introduction to Variational Autoencoders." *CoRR* abs/1906.02691. http://arxiv.org/abs/1906.02691.

Kröse, Ben, Ben Krose, Patrick van der Smagt, and Patrick Smagt. 1993. "An Introduction to Neural Networks."

Leung, Henry, and Simon Haykin. 1991. "The Complex Backpropagation Algorithm." *IEEE Transactions on Signal Processing* 39 (9): 2101–4.

Masegosa, AndrÃ©s R., Rafael Cabañas, Helge Langseth, Thomas D. Nielsen, and Antonio Salmerón. 2019. "Probabilistic Models with Deep Neural Networks." http://arxiv.org/abs/1908.03442.

Mayranna. 2013. *Wikimedia.org*. https://commons.wikimedia.org/w/index.php?curid=30128320.

Pan, Zhaoqing, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. 2019. "Recent Progress on Generative Adversarial Networks (Gans): A Survey." *IEEE Access* 7: 36322–33.

Salimans, Tim, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. "Improved Techniques for Training Gans." *CoRR* abs/1606.03498. http://arxiv.org/abs/1606.03498.

Serrano, Luis. 2017. "A Friendly Introduction to Convolutional Neural Networks and Image Recognition." *YouTube*. https://www.youtube.com/watch?v=2-Ol7ZB0MmU.

Wei, Ruoqi, Cesar Garcia, Ahmed El-Sayed, Viyaleta Peterson, and Ausif Mahmood. 2020. "Variations in Variational Autoencoders-a Comparative Evaluation." *IEEE Access* 8: 153651–70.

Zeldes, Yoel. 2018. "Neural Networks from a Bayesian Perspective." *Medium*. Towards Data Science. https://towardsdatascience.com/neural-networks-from-a-bayesian-perspective-ad8cacc7588

2020. *AI Wiki*. https://docs.paperspace.com/machine-learning/wiki/activation-function.